Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería en Tecnologías de la Telecomunicación

Trabajo de Fin de Grado

Estudio de sistemas inerciales en el seguimiento de terapias rehabilitadoras basadas en Machine Learning

Autora: Andrea Martínez Parra

Tutora: Dña. Ana Jiménez Martín

Cotutora: Dña. Sara García de Villa

Curso 2020/2021

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería en Tecnologías de la Telecomunicación

Trabajo de Fin de Grado

Estudio de sistemas inerciales en el seguimiento de terapias rehabilitadoras basadas en Machine Learning

Autora: Andrea Martínez Parra

Tutora: Dña. Ana Jiménez Martín

Cotutora: Dña. Sara García de Villa

Tribunal:

Presidente:	D. Felipe Espinosa Zapata
Vocal 1**:**	D. Fernando Naranjo Vega
Vocal 2**:**	Dña. Ana Jiménez Martín

Fecha: 24 de febrero de 2021

"El propósito de la educación es mostrar a la gente cómo aprender por sí mismos."

Noam Chomsky

Agradecimientos

En primer lugar agradecer a mi tutora Ana y a mi cotutora Sara por ayudarme, aconsejarme y guiarme en la realización de este proyecto. Han sido muchas horas de grabación de ejercicios y sin su apoyo, sin la constancia y eficiencia de sus consejos este proyecto no hubiera salido a la luz.

Dar las gracias a todas las personas que empezaron siendo compañeros de carrera pero ahora son amigos increíbles, ha sido un placer poder haber tenido la oportunidad de crecer todos juntos, espero que esta unión jamás se rompa. Sobre todo a quien quiero agradecer es a María, mi gran descubrimiento en la carrera, increíble compañera en todos los aspectos, nos hemos pasado horas y horas haciendo laboratorios, redactando memorias, estudiando exámenes imposibles, sin ella la carrera hubiera sido muy complicada de terminar. Irene y Marta también han sido unos pilares fundamentales en estos años, gracias por confiar en mí y por estar siempre a mi lado disfrutando de este viaje, sois increíbles. A Alex, Richi, Carlos, Borja y Laura por tantos momentos inigualables. Ha sido un placer disfrutar con vosotros de esta carrera tan complicada pero a la vez tan bonita.

También quiero agradecer a mis amigos de toda la vida por estar siempre a mi lado celebrando todo lo bueno que nos ofrece la vida. A María, Inés, Álvaro y Mañas, habéis sido, sois y seréis los mejores compañeros de aventuras. Mil gracias por todo.

El agradecimiento más especial va dirigido a toda mi familia, la que ha estado siempre presente aconsejándome y ayudándome a seguir luchando y a no rendirme en estos años tan duros de estudio. Gracias Mamá, Papá y Jorge sin vosotros hubiera sido imposible que este largo viaje llegara a su fin.

Por último, quiero dar las gracias a Ian, no puedo tener más suerte de tener a una persona como él a mi lado, apoyándome en todo momento y confiando en mí siempre. No puedo estar más agradecida, mil gracias por estar siempre, sin él no hubiera sido posible conseguir todo lo que he conseguido.

Resumen

Este trabajo ha desarrollado y caracterizado una herramienta para monitorizar ejercicios físicos de terapias pautadas empleando los datos obtenidos de cuatro unidades de medida inercial (IMUs). La monitorización incluye la identificación del ejercicio entre un catálogo y su evaluación, entre bien o mal. Dicha clasificación se ha realizado mediante algoritmos de Machine Learning. Para este fin, se optimiza la posición y el número de IMUs empleadas. Además, se determina K-Nearest Neighbours como el clasificador más adecuado y el número de IMUs óptimo en dos, una por extremidad. Con ello, se obtienen exactitudes en identificación y evaluación del $99,5\,\%$.

Palabras clave: IMU; Algoritmos; Detección; Machine Learning; UKF.

Abstract

This work has developed and characterized a tool to monitor physical exercises of paused therapies using the data obtained from four units of inertial measurement (IMUs). Monitoring includes identifying the exercise between a catalog and evaluating it, right or wrong. This classification was done using Machine Learning algorithms. For this purpose, the position and number of IMUs used is optimized. In addition, K-Nearest Neighbours is determined as the most suitable classifier and the optimal number of IMUs in two, one per limb. This results in accuracies in identification and evaluation of 99,5 %.

Keywords: Inertial Sensors (IMU); Algorithms; Detection; Machine Learning; UKF.

Resumen extendido

El envejecimiento de la población en países desarrollados ha dado lugar a un incremento en la demanda de servicios sanitarios y sociales asociados a estos adultos mayores. Esto ha dado lugar a un interés creciente por tener un envejecimiento saludable que permita tener una vida plena e independiente el mayor tiempo posible. Esto implica comida sana y ejercicio diario, por lo que cada vez más se pautan tablas de ejercicios específicos a los adultos mayores. Actualmente hay multitud de dispositivos y aplicaciones que permiten monitorizar la actividad a través de la estimación del número de pasos utilizando una Unidad de Medida Inercial (IMU). Sin embargo, no hay muchos trabajos que aborden la monitorización de ejercicios como tal. Esto tiene una doble finalidad, por un lado, sirve de ayuda al paciente, haciendo las veces de un entrenador virtual y, por otro lado, a los cuidadores o facultativos les permite evaluar el desempeño de los ejercicios físicos realizados.

El principal objetivo de este Trabajo de Fin de Grado (TFG) es desarrollar una herramienta capaz de identificar y evaluar ejercicios físicos repetitivos de ejercicios pautados a partir de los datos recogidos en varias IMUs. El uso de sistemas inerciales simplifica el sistema al ser sensores pequeños, fáciles de utilizar y de bajo coste. Se utilizan cuatro sensores tanto para el seguimiento del tren inferior como el superior. La clasificación se realiza entre los ejercicios recogidos en un catálogo que engloba tres ejercicios centrados en ejercitar las extremidades inferiores, tres con las superiores y evaluación de la marcha.

La clasificación, se ha llevado a cabo utilizando métodos de Machine Learning supervisados. En concreto, se han evaluado los clasificadores de Naive Bayes, Support Vector Machines, K-Nearest Neighbours y Random Forest. Las entradas a los clasificadores de Machine Learning han sido cuatro características extraídas en los diferentes segmentos obtenidos mediante una ventana deslizante de cada una de las señales del seguimiento de los movimientos. Las características empleadas son la media, la desviación estándar, el máximo y el mínimo de cada segmento de la señal.

Se ha estudiado la influencia de las características de los ángulos de movimiento como entradas a los clasificadores. Para ello, primero se emplean únicamente las cuatro características obtenidas de las señales en crudo de las IMUs, es decir, de la velocidad angular y la aceleración lineal, y en el segundo lugar se amplía el número de características de entrada, considerando también las características obtenidas de los ángulos de Euler.

Se han abordado tres planteamientos distintos: uno donde solo se evalúa si un ejercicio concreto está bien o mal hecho, otro donde a partir de los ejercicios realizados solo identifica el tipo de ejercicio de entre el catálogo bajo estudio y por último y más general, que identifica y evalúa los ejercicios. En todos los planteamientos se emplean el 75 % de los datos para el entrenamiento de los clasificadores y el 25 %

restante se reserva únicamente para el test de los mismos. Además, se realiza una validación cruzada aleatoria con 10 iteraciones en la que los datos de entrenamiento y test cambian cada vez para asegurar la independencia de los datos de entrada.

El estudio se realizó sobre una base de datos de 14 pacientes que realizaron 4 series de cada uno de 7 ejercicios evaluados. Cada una de las series incluía entre 15 y 20 repeticiones, según la dificultad del ejercicio correspondiente, incluyendo movimientos tanto bien como mal realizados.

La evaluación de las diferentes clasificaciones se realiza sobre los datos de test. En dichas clasificaciones se evalúa la calidad de los ejercicios determinando si se realizan correcta o incorrectamente, obteniendo una sensibilidad del $99,5\,\%$. Además, también se evalúa la identificación de los ejercicios entre un catálogo propuesto, con una exactitud del $99,8\,\%$. Para todas estas clasificaciones el algoritmo que mejor funciona ante los datos de los ejercicios es el K-Nearest Neighbours y el número de IMUs óptimas es dos, situadas en las posiciones cruzadas o paralelas de las extremidades, inferiores o superiores.

Con respecto al estudio de las características de entrada, donde se evalúa la influencia de emplear los ángulos de Euler, los resultados obtenidos con las características obtenidas a partir de las medidas de las IMUs admiten un margen pequeño de mejora. Con ello, se observa que el introducir los ángulos de Euler resulta en una variación poco significativa, siempre menor al $1\,\%$, causada por la nueva aleatorización de los datos tras introducir más características. Por lo tanto, en este caso concreto no parece necesario incrementar la complejidad del sistema con nuevas características puesto que los resultados ya son satisfactorios.

Índice

Lista de figuras

Lista de tablas

Glosario

Lista de símbolos matemáticos

(X, Y, Z)	Coordenadas cartesianas del sistema de referencia fijo.
$(X^{''}, Y^{''}, Z^{''})$	Coordenadas cartesianas del sistema de referencia móvil.
ϕ	Roll. Ángulo de giro sobre el eje X.
θ	Pitch. Ángulo de giro sobre el eje Y.
ψ	Yaw. Ángulo de giro sobre el eje Z.
R_ψ	Matriz de rotación asociada al ángulo Yaw.
R_θ	Matriz de rotación asociada al ángulo Pitch.
R_ϕ	Matriz de rotación asociada al ángulo Roll.
M	Matriz de rotación total del sistema asociada a los ángulos de Euler.
$\vec{q}$	Cuaternión.
$\vec{q}_1$	Cuaternión asociado al giro sobre el eje Z.
$\vec{q}_2$	Cuaternión asociado al giro sobre el eje Y.
$\vec{q}_3$	Cuaternión asociado al giro sobre el eje X.
(a, b, c, d)	Coordenadas del cuaternión.
R^{-1}	Matriz de rotación total del sistema asociada al cuaternión.
$\vec{g}$	Fuerza de la gravedad.
$\vec{\omega}$	Velocidad angular
$\vec{a}$	Aceleración lineal
$\vec{f}$	Fuerza específica

Lista de acrónimos

IMU	Unidad de Medida Inercial (Inertial Measurement Unit)
ROC	Característica Operativa del Receptor (Receiver Operating Characteristic)
CV	Validación cruzada (Cross validation)
UKF	Filtro de Kalman *unscented* (*unscented* Kalman filter)
EKF	Filtro de Kalman *extended* (*extended* Kalman filter)
NB	Algoritmo de Naive Bayes
NB_k	Algoritmo de Naive Bayes con kernel

SVM	Algoritmo de Support Vector Machines
SVM_l	Algoritmo de Support Vector Machines con kernel lineal
SVM_g	Algoritmo de Support Vector Machines con kernel gaussiana
SVM_p	Algoritmo de Support Vector Machines con kernel polinomial
RF	Algoritmo de Random Forests
KNN	Algoritmo de K-Nearest Neighbours

Capítulo 1

Introducción

1.1 Motivación

La rehabilitación es un proceso fundamental tras la mayoría de las cirugías, ya que disminuye el tiempo de recuperación y evita problemas secundarios, lo que reduce las visitas innecesarias al médico o la prescripción de medicamentos. Sin embargo, ese sentimiento no suele ser compartido por los pacientes cuando realizan la terapia de forma independiente en su domicilio, debido a que les cuesta ser constantes y abandonan el tratamiento antes de lo recomendado. Esto no ocurre en la terapia presencial, pero supone un aumento de la atención médica y, por lo tanto, del coste sanitario. Una alternativa a la rehabilitación presencial es la monitorización de los ejercicios que realiza el paciente. Se sabe, en el ámbito de la fisioterapia, que la rehabilitación remota es posible y que tiene resultados equiparables a las valoraciones y tratamientos presenciales.

El sistema de referencia para la monitorización de movimiento del cuerpo humano es el denominado MoCap (Motion Capture), basado en la grabación mediante cámaras. Este es ampliamente utilizado en estudios de optimización de rendimiento deportivo, rehabilitación o animación 3D. Sin embargo, estos sistemas son costosos, no son portables y requieren de personal cualificado, lo que les descarta para el seguimiento de terapias de rehabilitación en el domicilio.

Uno de los grandes avances en el ámbito de la rehabilitación es el uso de los sensores portables como guía en la rehabilitación remota ya que ofrecen una alternativa mucho más económica y práctica que el MoCap. Los sensores utilizados son los sistemas de medida inerciales (IMU), que combinan acelerómetros, giróscopos y magnetómetros. Gracias a su tamaño, pueden ubicarse directamente sobre los segmentos específicos que se deseen monitorizar. A partir de la estimación de la cinemática de las articulaciones se puede hacer el seguimiento del ejercicio. Sin embargo, la medida no es directa por lo que es necesario generar algoritmos específicos para la aplicación concreta que se desee.

Para poder obtener esos algoritmos, se necesitan estudiar las respuestas de las señales obtenidas de los sensores, haciendo uso de métodos de aprendizaje supervisados, como determinados algoritmos de Machine Learning.

Este Trabajo Fin de Grado se centra en realizar un clasificador que permita identificar el ejercicio realiza-

do, dentro de un catálogo, así como la evaluación de los mismos; entre ejercicios bien y mal ejecutados. Lo que da lugar a un análisis de los datos a la hora de llevar a cabo un análisis del proceso de rehabilitación de un individuo.

A parte del interés del proyecto por su temática, hay otras motivaciones que han impulsado al desarrollo del mismo, como el trabajar en el análisis de datos a partir de herramientas de obtención y procesamiento de datos, adquirir habilidades relacionadas con la búsqueda de soluciones a problemas reales y enfrentarse a los problemas con actitud crítica.

El proyecto propuesto se ha desarrollado íntegramente en el seno del grupo de investigación GEINTRA de la Universidad de Alcalá. El grupo tiene una dilatada experiencia en temas de localización y posicionamiento lo que ha facilitado el uso de los sistemas inerciales.

1.2 Objetivos

El objetivo fundamental del proyecto es la identificación de ejercicios de entre un catálogo propuesto y la evaluación de los mismos, es decir, si están bien o mal ejecutados. Para poder evaluar dichos ejercicios se hace uso de IMUs las cuales ofrecen la información, que procesada, se introduce en los algoritmos de Machine Learning para clasificarla. Para cumplir con este objetivo principal será necesario alcanzar los siguientes objetivos parciales:

- Estimación precisa de los ángulos relativos de los segmentos monitorizados a partir de los datos recogidos por las IMUs.
- Clasificación del ejercicio de entre un catálogo registrado previamente.
- Verificación de la correcta ejecución de los ejercicios en base a unos parámetros definidos previamente.

El diagrama de bloques mostrado en la Figura 1.1 representa la estructura general del sistema a desarrollar. Primero, se realizan los diferentes tipos de movimientos que se quieran estudiar, las IMUs detectan esos movimientos y generan unos archivos con los datos de velocidad angular ($\vec{\omega}$) y aceleración lineal o fuerza específica ($\vec{f}$). Con los datos obtenidos de las IMUs, se estiman los ángulos de movimiento de dicha extremidad, empleando ángulos de Euler o cuaterniones. Utilizando la información de los ángulos, se determina el tipo de movimiento que se está realizando dentro de un catálogo de posibles movimientos. Por último, se evalúa el mismo, es decir, qué ejercicio se está realizando y la calidad del mismo, a partir de unos parámetros previamente definidos.

1.3 Estructura

El proyecto está estructurado en cinco capítulos principales que se explican a continuación:

- **Capítulo 1 - Introducción:** En este capítulo se hará una breve descripción del problema planteado, es decir, el problema en la rehabilitación y la posibilidad de ser remota mediante el uso de

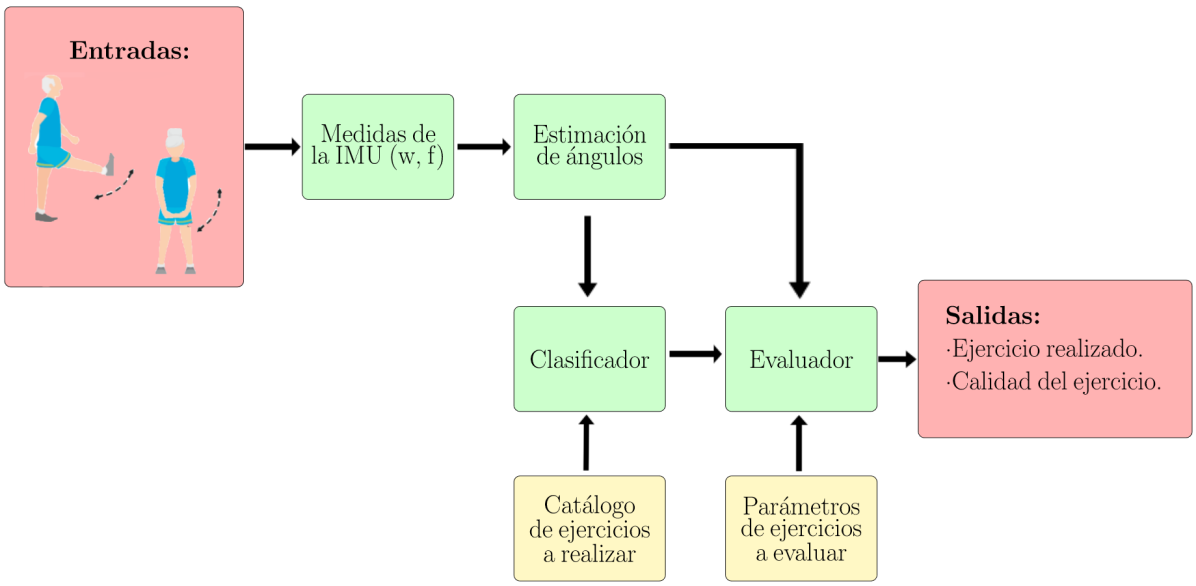


Figura 1.1: Diagrama de bloques del proceso de seguimiento de la monitorización de los movimientos de diferentes segmentos del cuerpo humano a través del uso de IMUs.

sensores de medida inercial. Además se explican la motivación y los objetivos que han influido en la realización de este proyecto.

- **Capítulo 2 - Estado del arte:** Se detalla en profundidad los estudios en los que se han hecho uso de las IMUs a la hora de detectar y analizar ejercicios relacionados con la actividad física. Además se explica matemáticamente las IMUs, la obtención de la orientación y los algoritmos de Machine Learning utilizados para la clasificación de los ejercicios.

- **Capítulo 3 - Desarrollo:** Se explican los ejercicios elegidos para monitorizar la rehabilitación de una persona, las posición sobre las piernas y los brazos en los que se tienen que situar las IMUs. Además se determina el proceso de grabación y sincronización de los ejercicios para poder obtener las características obtenidas a partir de la segmentación por ventana deslizante. Por último se determinan los tipos de clasificación utilizados.

- **Capítulo 4 - Resultados:** Se representan los resultados obtenidos al realizar las clasificaciones definidas en el apartado de desarrollo. Además, se realiza una discusión en la que se determina cuál es el algoritmo que mejor funciona para la detección y evaluación de los ejercicios así como las posiciones óptimas a la hora de situar las IMUs sobre las extremidades corporales.

- **Capítulo 5 - Conclusiones y trabajos futuros:** Se realiza un resumen de las conclusiones a las que se ha llegado en el apartado de resultados y se exponen las propuestas futuras.

Capítulo 2

Estado del arte

Con el fin de garantizar la eficacia adecuada de los programas de fisioterapia, hay una necesidad de la evaluación sistemática en la ejecución de ejercicios prescritos. Investigaciones recientes exploran las oportunidades derivadas de la tecnología para proporcionar mejoras en la supervisión de los ejercicios de fisioterapia en el hogar. Fuera del entorno clínico, los sistemas se basan en la recuperación de datos de movimiento, cuantificar adecuadamente y caracterizar estos datos y emitir recomendaciones al paciente. Los métodos para lograr un análisis biomecánico consisten en plataformas de fuerza y equilibrio, sistemas de captura de movimiento basados en la visión y sistemas basados en sensores inerciales.

La tecnología de los sensores inerciales o IMUs ha avanzado ya que son accesibles y muy empleadas. Dichos sistemas inerciales están compuestos por acelerómetros, giróscopos y magnetómetros los cuales son capaces de medir la aceleración lineal, la velocidad angular y el campo magnético, respectivamente. Los sensores inerciales son una de las alternativas disponibles para lograr un análisis de movimiento remoto, ya que son portables, ligeros y rentables. Pueden ser totalmente autónomos en el usuario para que puedan moverse libremente durante el análisis y la grabación, lo que posibilita la realización de los movimientos en lugares que no sean clínicas o entornos cerrados.

Además, los sensores portables, como las IMUs, ya se han utilizado para una amplia variedad de aplicaciones de análisis de movimiento. Varios estudios se han centrado en el análisis de la marcha humana [1] utilizando IMUs, las cuales se usan para medir la calidad de la ejecución del movimiento de caminar mediante la estimación y su posterior comparación con la señal de los ángulos que debería haber si se camina normal. Las mediciones al caminar capturadas con las IMUs también se pueden utilizar como entrada para dispositivos de asistencia de extremidades inferiores, como prótesis robóticas y exoesqueletos. En [2] se hace uso de la medición de los ángulos obtenidos a partir de la articulación de la rodilla para poder detectar los movimientos de las extremidades inferiores en los ancianos o en personas que poseen problemas de movilidad. Sus resultados concluyen que la medición de dichos ángulos da lugar a una clasificación de los movimientos de caminar en las extremidades inferiores. Además, se han utilizado ampliamente para el análisis cinemático de las extremidades superiores [3, 4] ya que dichos sensores son capaces de medir la velocidad, rotación y aceleración de los movimientos haciendo posible el estudio de los movimientos de los brazos para así asegurar la buena ejecución de los mismos. También se utilizan en procesos de rehabilitación [1], ya que se pueden situar sobre las extremidades para así hacer un segui-

miento de posibles complicaciones a la hora de ejecutar los movimientos o realizar algún tipo de ejercicio físico. En este estudio, se sitúan sobre la cadera del paciente para estudiar la postura de la espalda a la hora de realizar diferentes movimientos como andar o para tratar problemas de postura.

Para evaluar la calidad del ejercicio, en algunos estudios se ha investigado la viabilidad de los sensores inerciales para proporcionar una clasificación precisa del rendimiento del ejercicio en pacientes que ejecutan ejercicios de miembros inferiores y superiores para seguimiento de la rehabilitación [5, 1]. En dichos estudios se observa que al situar las IMUs en los muslos y en las espinillas y en los brazos y antebrazos de los pacientes se puede determinar casi al 100 % la calidad del ejercicio, es decir, si está bien hecho o mal hecho. En la elevación de pesos [6] en la que se explica la utilización de los sensores inerciales para la detección y caracterización de la elevación de pesos, aunque también son capaces de detectar la postura, la dirección de giro y el movimiento vertical con una exactitud que ronda el 98 %. Además, también se investiga la viabilidad de dichos sensores a la hora de detectar los movimientos realizados en el proceso de entrenamiento [3] a partir de las señales ofrecidas por las IMUs son capaces de distinguir qué ejercicios se están haciendo de entre un catálogo que se ha propuesto. En [3] se evalúa la calidad de diferentes ejercicios de fisioterapia con las IMUs añadiendo sensores EMG que se encargan de evaluar la salud de los músculos y las células nerviosas que los controlan.

Para la detección y evaluación de los ejercicios realizados se hace uso de algoritmos de Machine Learning que ayudan a la identificación y evaluación de dichos ejercicios. D. Whelan et al. [7] propone, para la detección de los ejercicios de zancada, trabajar con los algoritmos de Naive Bayes (NB) y Support Vector Machines (SVM) con kernel gaussiana. E. Preatoni et al. [3] defiende que para detectar ejercicios de entrenamiento hace uso del algoritmo SVM y el K-Nearest Neighbours (KNN). A la hora de clasificar sentadillas con una pierna para la rehabilitación de la rodilla R. Kianifar et al. [8] propone hacer uso de los algoritmos SVM, KNN y Random Forests (RF). Además, en la evaluación de los ejercicios del proceso de rehabilitación que explica A. Pereira et al. [5] se hace uso de los algoritmos de KNN, SVM y RF.

En este trabajo se analizan los ejercicios de rehabilitación incluidos dentro de unas rutinas de específicas estudiando tanto la identificación del tipo de ejercicio realizado como la evaluación de los mismos. Para estudiar los diferentes ejercicios se realiza una clasificación de la información de la aceleración lineal y velocidad angular producida por las IMUs. Además, se estudia el efecto del estudio de los ángulos de Euler en estas clasificaciones, realizando otra clasificación en la que se añade la información de los ángulos de giro de dichas IMUs. En este análisis se estudia el mejor algoritmo de Machine Learning para esta aplicación, en la que se consideran en primer lugar la evaluación de los ejercicios individuales, la identificación del conjunto de los mismos y, por último, el planteamiento de identificación y evaluación de estos movimientos. Asimismo, se evalúan diferentes posiciones de IMUs con el fin de determinar el menor número de dispositivos para la aplicación.

2.1 Unidades de medida inercial: IMUs

Una IMU es un dispositivo electrónico que mide el movimiento de un objeto. Normalmente consta de tres giróscopos y tres acelerómetros que miden la velocidad angular y la aceleración lineal con influencia de la gravedad, respectivamente. Además se incluyen magnetómetros los cuales miden la fuerza total del campo

magnético. La limitación que ofrecen los magnetómetros es que son muy sensibles a las interferencias electromagnéticas lo que puede dar lugar a un incremento en el error en su medida. Este tipo de unidades implementan internamente tres ejes ortogonales sobre los cuales se montan los sensores uniaxiales, de manera que a cada eje se le asigna un acelerómetro y un giróscopo, tal y como se puede observar en la Figura 2.1.

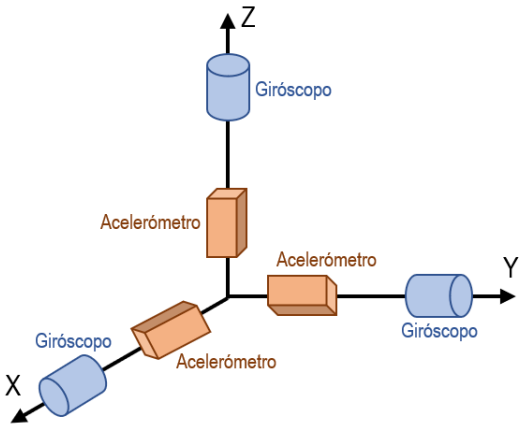


Figura 2.1: Acelerómetros y giróscopos en los tres ejes ortogonales de una IMU.

Para conocer correctamente el funcionamiento de las IMUs, se explica con detalle la función y el funcionamiento de los acelerómetros y los giróscopos. En este proyecto no se hace uso de los magnetómetros ya que son muy sensibles a interferencias electromagnéticas, que dan lugar a unas medidas erróneas del campo magnético ofrecidas por el magnetómetro.

ACELERÓMETRO:

Los acelerómetros son sensores inerciales que basan su funcionamiento en la Segunda Ley de Newton. Proporcionan a su salida la medida de la segunda derivada de la posición del dispositivo sobre el que se encuentra, es decir, la aceleración lineal a la que está sometido el mismo.

La aceleración que mide el acelerómetro está compuesto por la aceleración lineal instantánea ($\vec{a}$) y la influencia de la fuerza de la gravedad ($\vec{g}$) a la que está sometida el dispositivo, la combinación de estas magnitudes da lugar a la fuerza específica ($\vec{f}$).

El acelerómetro está incluido en diferentes dispositivos usados en aplicaciones de uso doméstico, en teléfonos móviles, estabilización de cámaras, videoconsolas y un largo sin fin de aplicaciones.

GIRÓSCOPO:

El giróscopo es un dispositivo mecánico que sirve para medir el cambio de orientación en el espacio del objeto sobre el que se encuentra. Mide la velocidad angular $\vec{\omega}$ instantánea de un dispositivo, es decir, la cantidad de rotación del dispositivo sobre cada uno de los ejes. Esta velocidad angular es determinada a partir de un disco giratorio montado dentro de una serie de anillos de rotación [9].

Las aplicaciones típicas en las que se hace uso de los giróscopos son la robótica, en los sistemas de control de los vehículos, en los análisis de marcha humana y la animación.

2.1.1 Estimación de la orientación

La información suministrada por una IMU es la aceleración lineal y la velocidad angular correspondientes a cada uno de los ejes del sistema. A partir de dicha información, los ángulos de giro sobre el eje X e Y se pueden estimar a partir de la relación de la fuerza medida ($\vec{f}$) y el vector de la gravedad ($\vec{g}$). Dichas estimaciones proveen un error o deriva, conocido como *bias* lo que da lugar a un error sistemático en las medidas integradas. Para disminuir este error en la orientación, se fusiona la información de todos los sensores de la IMU.

En la estimación de la orientación a partir de magnitudes inerciales se fusionan las medidas de los acelerómetros y giróscopos para obtener la orientación del sensor. Una herramienta común para realizar esta estimación son los **filtros de Kalman**. Los filtros de Kalman son filtros bayesianos que se utilizan para estimar los estados actuales de un sistema a partir del estado del sistema en el momento anterior y la información de las medidas de los sensores en sistemas lineales, existiendo sus adaptaciones a sistemas no lineales, como son el filtro de Kalman extendido (EKF) y el filtro Kalman "*unscented*" (UKF). Estos filtros combinan una etapa de predicción y otra de actualización. En la etapa de predicción se implementa el modelo del sistema, en la que representa cómo evolucionan en el tiempo los estados del sistema y, en la de corrección, se actualizan los estados estimados a partir de las mediciones de los sensores.

En [10], se propone un algoritmo de estimación de los ángulos de Euler basado en un UKF, cuya etapa de predicción se basa en la integración en el tiempo de las mediciones de la velocidad angular para estimar los ángulos y el bias del giróscopo. La etapa de actualización incorporan en determinados instantes de tiempo t las medidas de aceleración, con el fin de reducir el error debido al ruido y a la integración en el tiempo de las medidas de la velocidad de giro que contienen las *bias*.

En los momentos en los que la aceleración debida al movimiento es nula, sólo se mide sobre el acelerómetro la influencia de la fuerza de la gravedad ($\vec{g}$), por lo que se puede obtener una referencia de orientación absoluta a partir de las medidas del acelerómetro. En este caso, los ángulos de giro sobre el eje X y el eje Y, denominados como Roll (ϕ) y Pitch (θ) respectivamente, se pueden extraer en el momento t según las ecuaciones (2.1) y (2.2).

$$\phi^t = \arctan(\frac{g_Y^t}{g_Z^t}) \tag{2.1}$$

$$\theta^t = \arctan(\frac{-g_Z^t}{\sqrt{(g_Y^t)^2 + (g_Z^t)^2}}) \tag{2.2}$$

Siendo g_Y y g_Z la proyección de la fuerza de gravedad sobre los ejes Y y Z de la IMU, respectivamente.

2.1.2 Ángulos de Euler

Para poder representar los ángulos de giro sobre los ejes tridimensionales, es necesario conocer el funcionamiento de los ángulos de Euler. Dichos ángulos forman un conjunto de tres coordenadas angulares

que sirven para determinar la orientación de un sistema de referencia, normalmente móvil, con respecto a otro sistema de referencia fijo.

Se representa un ángulo como la secuencia de tres ángulos, cada uno de ellos girado sobre un determinado eje siguiendo siempre un orden preestablecido: el ángulo de giro sobre el eje Z se define como ψ (Yaw), el ángulo de giro sobre el eje Y es θ (Pitch), y el ángulo de giro sobre el eje X es ϕ (Roll). Estos ángulos de giro se establecen en sentido antihorario, considerando un sistema de coordenadas dextrógiro. La rotación de estos ángulos sobre sus correspondientes ejes se muestra en la Figura 2.2.

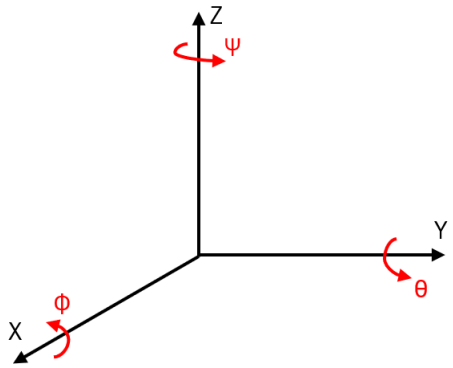


Figura 2.2: Ángulos de giro Roll (ϕ), Pitch (θ) y Yaw (ψ) representados en su sentido positivo.

Las matrices de rotación sobre un eje determinado representan los ángulos entre los ejes del sistema de referencia fijo y móvil al girar un cuerpo sobre un eje, el cual permanece constate. Estas matrices tienen siempre la misma estructura, mostrada en la ecuación (2.3).

$$R = \begin{pmatrix} \angle(X, X'') & \angle(X, Y'') & \angle(X, Z'') \\ \angle(Y, X'') & \angle(Y, Y'') & \angle(Y, Z'') \\ \angle(Z, X'') & \angle(Z, Y'') & \angle(Z, Z'') \end{pmatrix} \tag{2.3}$$

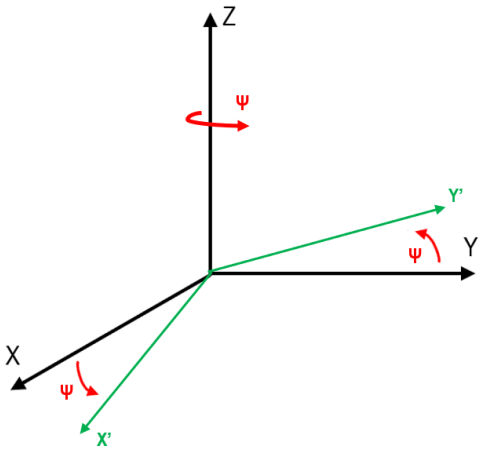
Siendo X, Y y Z los ejes del sistema de referencia fijo, y X'', Y'' y Z'' los del sistema de referencia móvil. Y donde $\angle(u, v)$ es el ángulo formado entre u y v.

En la primera fila de la matriz de rotación de la ecuación (2.3), se definen los ángulos creados entre el eje X del sistema de referencia fijo y los tres ejes del sistema de referencia móvil. Así mismo, la segunda fila representa el ángulo creado entre el eje Y del sistema de referencia fijo y los ejes del sistema de referencia móvil. Y, por último, la tercera fila representa el ángulo creado entre el eje Z del sistema de referencia fijo y los ejes del sistema de referencia móvil.

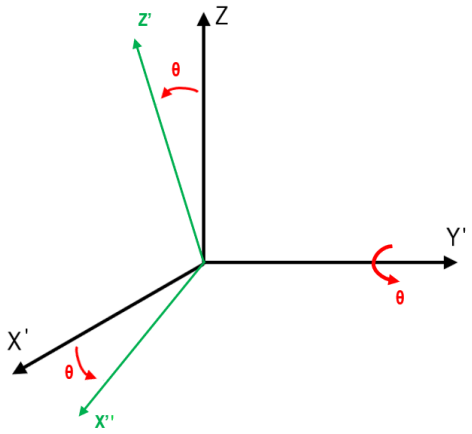
Habiendo definido las matrices de rotación sobre un eje, se explica cómo afectan los giros sobre cada eje en las matrices de rotación. Puesto que el orden de giros de los ángulos de Euler es ZYX, los giros, representados en la Figura 2.3, y sus matrices asociadas se explican en ese orden.

En la Figura 2.3a, se representa la rotación de un ángulo ψ sobre el eje Z, rotando los ejes X e Y que pasan a ser X' e Y'. La matriz de rotación de este giro sobre Z, R_ψ, se expresa en la ecuación (2.4).

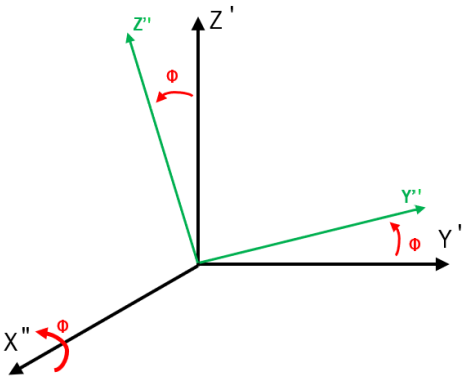
$$R_\psi = \begin{pmatrix} \cos(\psi) & \cos(90\,° + \psi) & \cos(90\,°) \\ \cos(90\,° - \psi) & \cos(\psi) & \cos(90\,°) \\ \cos(90\,°) & \cos(90\,°) & \cos(0\,°) \end{pmatrix} = \begin{pmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.4}$$



(a) Giro del ángulo ψ sobre el eje Z. Los ejes iniciales se muestran en negro mientras que los ejes finales son de color verde. El eje Z inicial y el Z final son el mismo.



(b) Giro del ángulo θ sobre el eje $Y^{'}$. Los ejes iniciales se muestran en negro mientras que los ejes finales son de color verde. El eje $Y^{'}$ inicial y el $Y^{'}$ final son el mismo.



(c) Giro del ángulo ϕ sobre el eje $X^{''}$. Los ejes iniciales se muestran en negro mientras que los ejes finales son de color verde. El eje $X^{''}$ inicial y el $X^{''}$ final son el mismo.

Figura 2.3: Giros de los ángulos ψ, θ y ϕ sobre los ejes Z, $Y^{'}$ y $X^{''}$, respectivamente.

Donde el ángulo entre X y $X^{'}$ es $\cos(\psi)$, el ángulo entre X e $Y^{'}$ es $\cos(90\,^\circ + \psi) = -\sin(\psi)$ y el ángulo entre X y $Z^{'} \equiv Z$ es $\cos(90\,^\circ) = 0$.

La Figura 2.3b representa el giro de un ángulo θ sobre el eje $Y^{'}$. Este giro crea los ejes secundarios $X^{''}$ y $Z^{'}$. La matriz de rotación asociada al ángulo θ, R_θ, viene definida en la ecuación (2.5).

$$R_\theta = \begin{pmatrix} \cos(\theta) & \cos(90\,^\circ) & \cos(90\,^\circ - \theta) \\ \cos(90\,^\circ) & \cos(0\,^\circ) & \cos(90\,^\circ) \\ \cos(90\,^\circ + \theta) & \cos(90\,^\circ) & \cos(\theta) \end{pmatrix} = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix} \tag{2.5}$$

En la Figura 2.3c se representa el giro de un ángulo ϕ sobre el eje $X^{''}$. Este giro crea los ejes secundarios

Y'' y Z'' y su matriz asociada, R_ϕ, viene determinada en la ecuación (2.6).

$$R_\phi = \begin{pmatrix} \cos(0\,^\circ) & \cos(90\,^\circ) & \cos(90\,^\circ) \\ \cos(90\,^\circ) & \cos(\phi) & \cos(90\,^\circ + \phi) \\ \cos(90\,^\circ) & \cos(90\,^\circ - \phi) & \cos(\phi) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{pmatrix} \tag{2.6}$$

Definidas las matrices de rotación asociadas a cada uno de los giros R_ψ, R_θ y R_ϕ en las ecuaciones (2.4), (2.5) y (2.6), se establece la matriz de rotación total M como la combinación de estas tres. M permite representar las coordenadas del sistema de referencia móvil con respecto del fijo, por lo que integra la matrices de rotación en cada eje invertidas R_ψ^{-1}, R_θ^{-1} y R_ϕ^{-1}, como representa la ecuación (2.7).

$$\begin{pmatrix} X'' \\ Y'' \\ Z'' \end{pmatrix} = R_\phi^{-1} \cdot R_\theta^{-1} \cdot R_\psi^{-1} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = M \cdot \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{2.7}$$

En la Figura 2.4 se presentan el sistema de referencia fijo (sistema en negro) y el sistema de referencia móvil (sistema en verde) girado y centrado en el origen del sistema de referencia fijo, el cual se define con la ecuación (2.7).

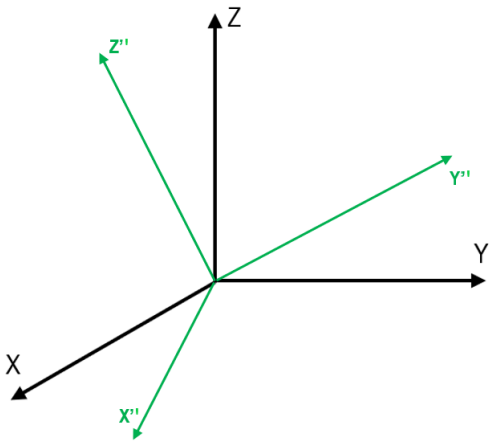
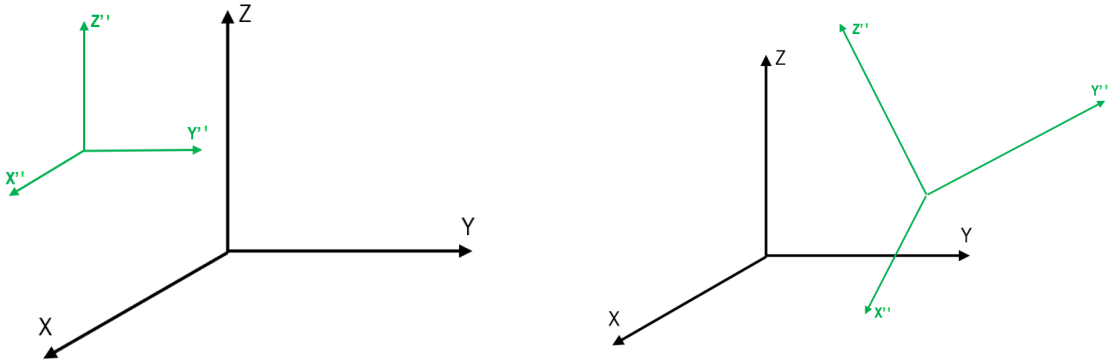


Figura 2.4: Sistema de referencia móvil girado con respecto al sistema de referencia fijo y centrado en el origen del mismo.

Siendo M (2.8) la matriz de rotación total del sistema, calculada a partir de la multiplicación de las matrices de rotación inversas de los ángulos Roll (ecuación (2.6)), Pitch (ecuación (2.5)) y Yaw (ecuación (2.4)).

$$M = \begin{pmatrix} \cos(\psi)\cos(\theta) & \sin(\psi)\cos(\theta) & -\sin(\theta) \\ -\sin(\psi)\cos(\phi) + \cos(\psi)\sin(\theta)\sin(\phi) & \cos(\psi)\cos(\phi) + \sin(\psi)\sin(\theta)\sin(\phi) & \cos(\theta)\sin(\phi)) \\ \cos(\psi)\sin(\theta)\cos(\phi) + \sin(\psi)\sin(\phi) & \sin(\psi)\sin(\theta)\cos(\phi) - \sin(\psi)\sin(\phi) & \cos(\theta)\cos(\phi) \end{pmatrix} \tag{2.8}$$

Existen otros tipos de de sistemas de referencia móviles, los cuales no siempre están centrados en el origen del sistema de referencia fijo. En la que se observa que el sistema de referencia móvil no está centrado en el origen del fijo pero sí mantiene la misma dirección que él. En cambio, en la Figura 2.5b el sistema de referencia móvil se encuentra girado y no centrado en el origen del sistema de referencia fijo.



(a) Sistema de referencia móvil en la misma dirección que el sistema de referencia fijo y no centrado en el origen del mismo.

(b) Sistema de referencia móvil girado con respecto al sistema de referencia fijo y no centrado en el origen del mismo.

Figura 2.5: Posibles posiciones del sistema de referencia móvil con respecto del sistema de referencia fijo.

Aunque los sistemas de referencia móvil de las Figuras 2.4, 2.5a y 2.5b estén orientados y centrados en diferentes posiciones, la ecuación para calcular las coordenadas de un vector con componentes X'', Y'' y Z'' con respecto a las coordenadas del sistema fijo es la misma para todos, es decir, se usa siempre la ecuación (2.7).

Los ángulos de Euler conforman la forma más usual de representar el movimiento de un sensor de medida inercial y determinan tanto la velocidad angular como la aceleración lineal. Aunque los ángulos de Euler suelen ser los que más se usan para representar movimientos, presentan algunos serios problemas. Si se aproxima el ángulo Pitch a 90 ° la estimación de los ángulos está indeterminada y se han de aproximar con asunciones de los ángulos θ o ψ. Para solventar esta limitación, existen otros tipos de representación de ángulos en el espacio, como son los cuaterniones.

2.1.3 Cuaterniones

Otra forma de entender la rotación es considerarla como una única rotación sobre un vector tetradimensional, cuyos elementos son función de ese vector y de la magnitud de la rotación. Los cuaterniones son una extensión de los números reales, generados a partir de la adición de las unidades imaginarias i, j y k.

Un cuaternión se puede representar de las tres formas posibles que se explican en la ecuación (2.9). En la primera de ellas, se representa un cuaternión como un punto en el espacio de cuatro dimensiones, en la segunda se representa en formato matricial y en la tercera se representa como un conjunto de un número real más un vector tridimensional $\vec{v}$.

$$\vec{q} = \underbrace{(a,b,c,d)}_{\text{Punto en } R^4} = \underbrace{aI + b\vec{i} + c\vec{j} + d\vec{k}}_{\text{Formato matricial}} = \underbrace{a + \vec{v}}_{\text{Vector en } R^3} \tag{2.9}$$

Donde a, b, c y d pertenecen a los números reales y los elementos I, $\vec{i}$, $\vec{j}$ y $\vec{k}$ son cuaterniones, siendo $\vec{q}$ la combinación lineal de números reales y cuaterniones.

Relación entre los cuaterniones y los Ángulos de Euler

La forma usada para representar los movimientos a partir de sensores de medida inerciales mediante los ángulos de Euler y de los cuaterniones. De esta forma se establece una relación entre ambos tipos de representación.

1. El cuaternión asociado al giro sobre el eje Z es el determinado en la ecuación (2.10).

$$\vec{q_1} = \Big(\cos(\psi/2), \quad 0, \quad 0, \quad \text{sen}(\psi/2)\Big) \tag{2.10}$$

2. El cuaternión asociado al giro sobre el eje Y es el determinado en la ecuación (2.11).

$$\vec{q_2} = \Big(\cos(\theta/2), \quad 0, \quad \text{sen}(\theta/2), \quad 0)\Big) \tag{2.11}$$

3. El cuaternión asociado al giro sobre el eje X es el determinado en la ecuación (2.12).

$$\vec{q_3} = \Big(\cos(\phi/2), \quad \text{sen}(\phi/2), \quad 0, \quad 0\Big) \tag{2.12}$$

El cuaternión total, que representa el ángulo de giro suma de las tres rotaciones parciales en torno a los respectivos ejes, se determina a partir de los cuaterniones asociados al giro sobre cada uno de los ejes. Hay que tener en cuenta que la combinación de cuaterniones no es una multiplicación de vectores, su ecuación final viene definida en la ecuación (2.13).

$$\begin{aligned}\vec{q} = (a,b,c,d) = \vec{q_3}\cdot\vec{q_2}\cdot\vec{q_1} = \begin{bmatrix}\cos(\psi/2)\\0\\0\\\sin(\psi/2)\end{bmatrix}\begin{bmatrix}\cos(\theta/2)\\0\\\sin(\theta/2)\\0\end{bmatrix}\begin{bmatrix}\cos(\phi/2)\\\sin(\phi/2)\\0\\0\end{bmatrix}\\ = \begin{bmatrix}\cos(\phi/2)\cos(\theta/2)\cos(\psi/2)+\sin(\phi/2)\sin(\theta/2)\sin(\psi/2)\\ \sin(\phi/2)\cos(\theta/2)\cos(\psi/2)-\cos(\phi/2)\sin(\theta/2)\sin(\psi/2)\\ \cos(\phi/2)\sin(\theta/2)\cos(\psi/2)+\sin(\phi/2)\cos(\theta/2)\sin(\psi/2)\\ \cos(\phi/2)\cos(\theta/2)\sin(\psi/2)-\sin(\phi/2)\sin(\theta/2)\cos(\psi/2)\end{bmatrix}\end{aligned} \tag{2.13}$$

Al realizar el resto de operaciones convenientes con los cuaterniones asociados a los giros, se llega a la matriz de rotación asociada al cuaternión, R^{-1}, definida en la ecuación (2.14).

$$R^{-1} = \begin{pmatrix}2a^2+2b^2-1 & 2(bc+ad) & 2(-ac+bd)\\ 2(bc-ad) & 2a^2+2c^2-1 & 2(ab+cd)\\ 2(ac+bd) & 2(-ab+cd) & 2a^2+2d^2-1\end{pmatrix} \tag{2.14}$$

Para determinar las coordenadas del giro de un sistema de referencia móvil con respecto a uno fijo a partir de cuaterniones, se usa la misma ecuación que se usaba en los ángulos de Euler. Esta ecuación viene definida en (2.15).

$$\begin{pmatrix}X''\\Y''\\Z''\end{pmatrix} = R^{-1}\begin{pmatrix}X\\Y\\Z\end{pmatrix} \tag{2.15}$$

Además, para determinar las coordenadas del cuaternión en función de los ángulos de giro de los ejes se iguala la matriz M descrita en los ángulos de Euler (ecuación (2.8)) y la matriz R^{-1} descrita en la ecuación (2.14).

Considerando que se tienen las coordenadas del cuaternión, se puede hacer la operación inversa, es decir, obtener los ángulos de giro a partir de las coordenadas del cuaternión. Estos ángulos de giro se determinan en las ecuaciones (2.16), (2.17), (2.18).

$$\phi = \arctan(\frac{2ab + 2cd}{2a^2 + 2d^2 - 1}) \tag{2.16}$$

$$\theta = \arcsin(2ac - 2bd) \tag{2.17}$$

$$\psi = \arctan(\frac{2bc + 2ad}{2a^2 + 2b^2 - 1}) \tag{2.18}$$

A partir de las coordenadas del cuaternión de giro se puede definir el movimiento completo de un sistema de medida inercial sin las indeterminaciones que ocurren con los ángulos de Euler. La desventaja de los cuaterniones es que son vectores de cuatro dimensiones y son bastante complicados de justificar y representar tridimensionalmente.

2.2 Métodos de clasificación

El Machine Learning o Aprendizaje Automático es una disciplina científica capaz de crear modelos que, a partir de la identificación de patrones en gran cantidad de datos, son capaces de resolver una tarea dada. Estos modelos son entrenados a partir de algoritmos que usan gran cantidad de datos. Al identificar dichos datos, el modelo es capaz de predecir comportamientos.

Los algoritmos de Machine Learning pueden clasificarse en tres grandes tipos: aprendizaje supervisado, aprendizaje no supervisado o aprendizaje por refuerzo. En este proyecto se va a hacer uso de algoritmos de clasificación basados en aprendizaje supervisado.

La clasificación supervisada es una de las tareas que se llevan a cabo más frecuentemente por los sistemas inteligentes. Los algoritmos de clasificación supervisados se basan en lo que se conoce como información de entrenamiento. Estos trabajan con datos etiquetados, intentando encontrar una función que, dadas las variables de entrada, les asigne la etiqueta correcta de salida. Una vez que se le ha proporcionado suficiente cantidad de datos, se introducen nuevos datos sin etiquetar que se clasifican acorde a lo aprendido.

El paso previo a aplicar un método de clasificación es la partición del conjunto de datos en dos subconjuntos más pequeños, los cuales se utilizan como entrenamiento y test. El subconjunto de datos de entrenamiento es utilizado para estimar los parámetros del modelo y el subconjunto de datos de test se emplea para comprobar el comportamiento del modelo estimado con datos desconocidos para el modelo.

Al aplicar un método de clasificación normalmente se cometen errores. Por ejemplo, si se emplean datos binarios (0 y 1), hay ceros que se clasifican como unos y unos que se clasifican como ceros. A partir de las clasificaciones correctas y erróneas se puede crear la matriz de confusión, que se utiliza para evaluar la bondad de un clasificador y se presenta en la Tabla 2.1.

Tabla 2.1: Siendo Y_i el valor real conocido e $\hat{Y}_i$ el valor estimado. TN (True Negative) es la predicción correcta del valor negativo y TP (True Positive) es la predicción correcta del valor positivo. En cambio, FP (False Positive) se trata de una predicción incorrecta en la que se ha predicho un valor positivo cuando, en realidad, es un valor negativo, y en FN (False Negative) se ha predicho un valor negativo cuando debería haber sido un valor positivo.

	$\hat{Y}_i = 0$	$\hat{Y}_i = 1$
$Y_i = 0$	TN	FP
$Y_i = 1$	FN	TP

A partir de los valores expuestos en la matriz de confusión de la Tabla 2.1, se pueden definir diferentes métricas que sirven para clasificar si un algoritmo cumple las características buscadas para su aplicación. Las métricas que se definen para todos los algoritmos evaluados son: matriz de confusión, error de clasificación, exactitud, especificidad y sensibilidad.

Error de clasificación (Err): Indica el número de elementos clasificados erróneamente con respecto al total de elementos clasificados. Su ecuación viene definida en (2.19).

$$Err = \frac{FP + FN}{TN + FP + FN + TP} \tag{2.19}$$

Exactitud (Ex): Indica el número de elementos clasificados correctamente con respecto al total de elementos clasificados. Su ecuación viene definida en (2.20).

$$Ex = \frac{TN + TP}{TN + FP + FN + TP} = 1 - Err \tag{2.20}$$

Especificidad (E): Indica la cantidad de verdaderos negativos que el modelo ha clasificado en función del número total de valores negativos. Su ecuación se define en (2.21).

$$E = \frac{TN}{TN + FP} \tag{2.21}$$

Sensibilidad (S): Indica la cantidad de verdaderos positivos que el modelo ha clasificado en función del número total de valores positivos. Su ecuación se define en (2.22).

$$S = \frac{TP}{TP + FN} \tag{2.22}$$

Un método alternativo para evaluar los clasificadores es la **curva ROC** (Receiver Operating Characteristic). Se trata de una representación gráfica del rendimiento del clasificador que muestra la distribución de la sensibilidad frente a la especificidad.

En la Figura 2.6 pueden observarse posibles curvas ROC a obtener. La curva amarilla define la curva ROC ideal, en la que se representa un 100 % de sensibilidad (no tiene ningún falso negativo) y un 100 % de especificidad (no tiene ningún falso positivo). La curva roja representa una curva ROC buena si está más próxima a la ideal, o regular si está más próxima a la verde. Y, por último, la curva verde es la peor curva ROC que se puede obtener ya que en este caso, se puede determinar que ese clasificador es inservible para la aplicación que se está estudiando.

Como se ha explicado en la introducción del apartado los algoritmos de Machine Learning que más se utilizan en la literatura son: NB, NB_k, SVM, SVM_l, SVM_g, SVM_p, RF y KNN.

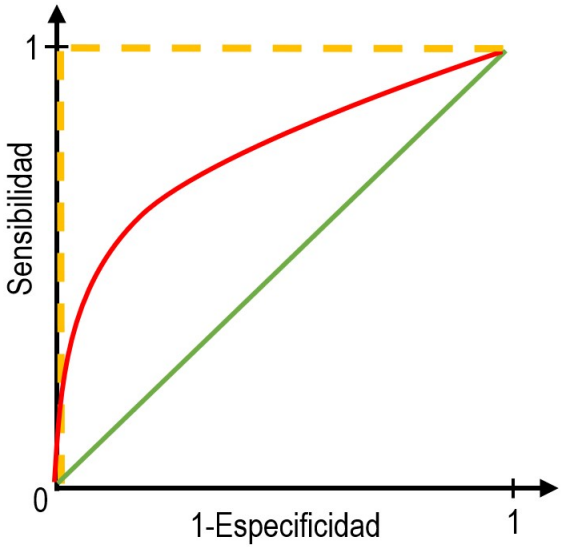


Figura 2.6: Tipos de curva ROC existentes.

2.2.1 Naive Bayes (NB)

El clasificador NB es un modelo probabilístico de Machine Learning que se usa generalmente en problemas de clasificación. Proporcionan una manera fácil de construir modelos con un comportamiento muy bueno debido a su simplicidad. En los algoritmos de NB se asume que las variables predictoras son independientes entre sí, es decir, que la presencia de una cierta característica en un conjunto de datos no está relacionada con la presencia de cualquier otra característica. Las matemáticas detrás de este tipo de clasificador son bastante fáciles de entender, y tiene un funcionamiento bastante bueno en multitud de casos.

El clasificador de NB está basado en el Teorema de Bayes, descrito en la ecuación (2.23), en la que $P(A|B)$ es la probabilidad condicional que ocurra el evento A cuando el evento B ya ha ocurrido; $P(B|A)$ es la probabilidad condicional que ocurra el evento B cuando el evento A ya ha ocurrido; y $P(A)$ y $P(B)$ son las probabilidades que ocurran los eventos A y B, respectivamente.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{2.23}$$

Para poder comprender su funcionamiento, se va a estudiar el ejemplo descrito en la Tabla 2.2.

Tabla 2.2: Ejemplo en el que se clasifica si el día es apto para jugar al golf dadas las características del día, denominados predictores.

Día	Pronóstico	Temperatura	Humedad	Viento	Jugar al golf
0	Lluvioso	Calor	Alta	No	No
1	Soleado	Leve	Alta	No	Sí
2	Soleado	Frío	Normal	Sí	No

Las columnas representan los predictores y las filas representan entradas individuales. Si se toma la primera fila del conjunto de datos, se puede observar que el día 0 no es apto para jugar al golf ya que el pronóstico es lluvioso, la temperatura es alta, la humedad es alta y no hay viento. A partir de esta

conclusión se pueden hacer dos suposiciones, una, como se indicó anteriormente, se considera que estos predictores son independientes, es decir, si la temperatura es alta, no significa necesariamente que la humedad sea alta. Otra suposición que se hace es que todos los predictores tienen el mismo efecto en el resultado, es decir, el día en el que hace viento no tiene más importancia a la hora de decidir si se puede jugar al golf o no.

El teorema de Bayes se puede reescribir según la ecuación (2.24).

$$P(y|X) = \frac{P(X|y) \cdot P(y)}{P(X)} \tag{2.24}$$

Siendo y la variable de clase, es decir, si es apto jugar al golf dadas las condiciones determinadas. Y la variable $X = (x_1, x_2, x_3, ..., x_n)$ representa los diferentes tipos de predictores (pronóstico, temperatura, humedad y viento). Al sustituir X en la ecuación (2.24) y expandirla haciendo uso de la regla de la cadena, se obtiene la probabilidad condicionada definida en la ecuación (2.25).

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)...P(x_n|y) \cdot P(y)}{P(x_1)...P(x_n)} \tag{2.25}$$

Se puede simplificar esta probabilidad condicionada ya que el denominador es común para todo el conjunto de datos. Es por ello que se puede introducir una proporcionalidad tal y como viene definido en la ecuación (2.26).

$$P(y|x_1, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y) \tag{2.26}$$

En el caso del ejemplo, la variable de clase y tiene dos resultados: sí o no, pero puede haber casos en los que la clasificación sea multivariante, por lo que hay que encontrar la clase y con máxima probabilidad haciendo uso de la ecuación (2.27).

$$y = \max_{y} P(y) \prod_{i=1}^{n} P(x_i|y) \tag{2.27}$$

Los algoritmos de NB se utilizan principalmente en clasificación de texto, análisis de opinión, filtrado de spam, sistemas de recomendación, etc. Son rápidos y fáciles de implementar y, además, funcionan correctamente en la predicción de clases múltiples. Es debido a esto que el algoritmo de NB se ha hecho uso en este proyecto ya que es bueno en la detección de los diferentes tipos de ejercicios así como su evaluación. Cuando se cumple la independencia entre predictores, un clasificador NB funciona mejor en comparación con otros modelos y necesita menos datos de entrenamiento. Sin embargo, tiene el inconveniente de que si existe una categoría que no se observó en el conjunto de datos de entrenamiento, el modelo asignará una probabilidad de cero y no podrá hacer una predicción. Aunque su mayor desventaja es el requisito de que los predictores sean independientes ya que en la mayoría de los casos de la vida real, los predictores son dependientes, lo que dificulta el desempeño del clasificador.

2.2.2 Support Vector Machines o Máquinas de Soporte Vectorial

SVM es un algoritmo de clasificación supervisado que puede resolver tanto problemas lineales como no lineales. El objetivo de las SVM es encontrar el hiperplano que clasifique los diferentes grupos de datos de entrada con el máximo margen en un espacio N-dimensional, denominado R_N. Estos datos de entrada

son vistos como un vector N-dimensional, el cual tiene un conjunto de puntos y estos puntos pueden pertenecer a dos posibles categorías. De esta manera, un algoritmo basado en SVM construye un modelo capaz de predecir si un punto nuevo pertenece a una categoría o a la otra.

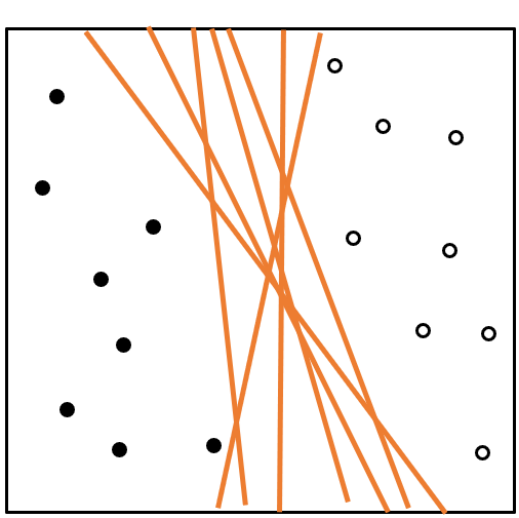
Los hiperplanos son límites de decisión con los que clasificar los datos. La ecuación del hiperplano se expresa en (2.28), donde el vector $\vec{w}$ se denomina vector normal, $\vec{x}$ es el vector de entrada y el escalar b se denomina intersección del hiperplano. El vector normal define la dirección del hiperplano, y la relación entre b y la norma del vector normal define la distancia entre el hiperplano y el origen del espacio.

$$< \vec{w}, \vec{x} > +b = 0 \longleftrightarrow \sum_{k-1}^{n} w_k x_k + b = 0 \tag{2.28}$$

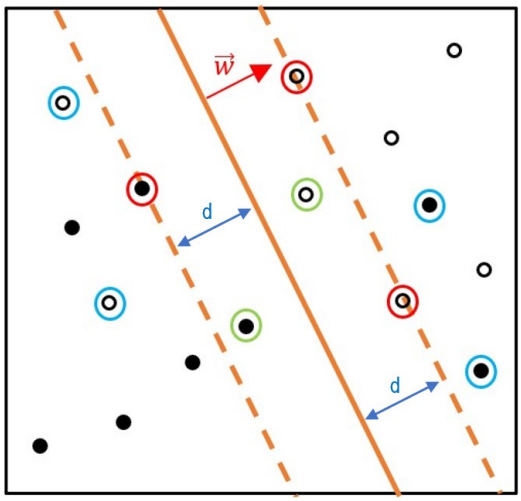
Donde $< \vec{w}, \vec{x} >$ indica el producto escalar de $\vec{w}$ y $\vec{x}$.

La dimensión del hiperplano depende del número de características. Si el número de características es dos, entonces el hiperplano es una línea. En cambio, si el número de datos de entrada es tres el hiperplano es un plano bidimensional.

Para separar las dos clases de puntos de datos, hay muchos hiperplanos posibles, como puede observarse en la Figura 2.7a. Las SVM buscan el hiperplano óptimo que tenga el margen máximo entre ambas clases. El margen m es la distancia d entre el hiperplano y una de las clases, más la distancia d entre el hiperplano y la otra clase, ambas presentadas en la Figura 2.7b. Maximizar este margen ayuda a que los puntos de datos futuros puedan clasificarse con mayor exactitud.



(a) Posibles hiperplanos de separación entre dos clases. La clase 0 es la rellena de negro y la clase 1 la rellena de blanco.



(b) Vectores de entrenamiento linealmente separados con una distancia d al hiperplano. El vector $\vec{w}$ es el vector normal y los vectores de soporte son los puntos rodeados con rojo.

Figura 2.7: Representación de los hiperplanos cuando existen dos clases de datos.

Los vectores de soporte (rodeados en rojo de la Figura 2.7b) son puntos de datos que están más cerca del hiperplano e influyen en la posición y orientación del mismo. Usando estos vectores de soporte, se maximiza el margen m del clasificador. Para poder definir correctamente el hiperplano, el mínimo de vectores de soporte es dos y el máximo es l, que es el tamaño del conjunto de entrenamiento ($\vec{x}$). La

distancia entre una clase y el hiperplano se define en la ecuación (2.29).

$$d = \frac{1}{||\vec{w}||} \tag{2.29}$$

Donde $||\vec{w}||$ denota la norma euclídea del vector $\vec{w}$.

Como se ha descrito anteriormente, el margen m es dos veces la distancia d entre una clase y el hiperplano, por lo que, a su vez, es dos veces la inversa de la norma $||\vec{w}||$ del vector normal (2.30)

$$m = d + d = \frac{2}{||\vec{w}||} \tag{2.30}$$

En el caso en el que los datos no sean linealmente separables, la clasificación de datos a partir del hiperplano no es perfecta. Es por ello que hay que introducir un término de error (ξ_i) en la determinación del hiperplano, que es proporcional a un valor C. Este valor C es una constante positiva cuyas principales metas son maximizar el margen y minimizar el número de errores en los datos de entrenamiento.

$$m = 2d + C\sum_{i=1}^{l}\xi_i = \frac{2}{||\vec{w}||} + C\sum_{i=1}^{l}\xi_i \tag{2.31}$$

El número de vectores de soporte viene afectado por C. Cuando se disminuye C el margen se expande y el número de vectores de soporte puede crecer.

Es muy común en los problemas reales clasificar un conjunto donde los ejemplos no suelen ser separables. Para resolver la clasificación mediante SVM de este tipo de conjuntos, se emplean funciones Kernel. Una función Kernel $k(\vec{x}, \vec{z})$ define implícitamente un espacio nuevo H, denominado espacio de características y un mapeo Φ, denominado mapeo de características, el cual transforma el espacio original R_N en H, tal y como se puede observar en la Figura 2.8.

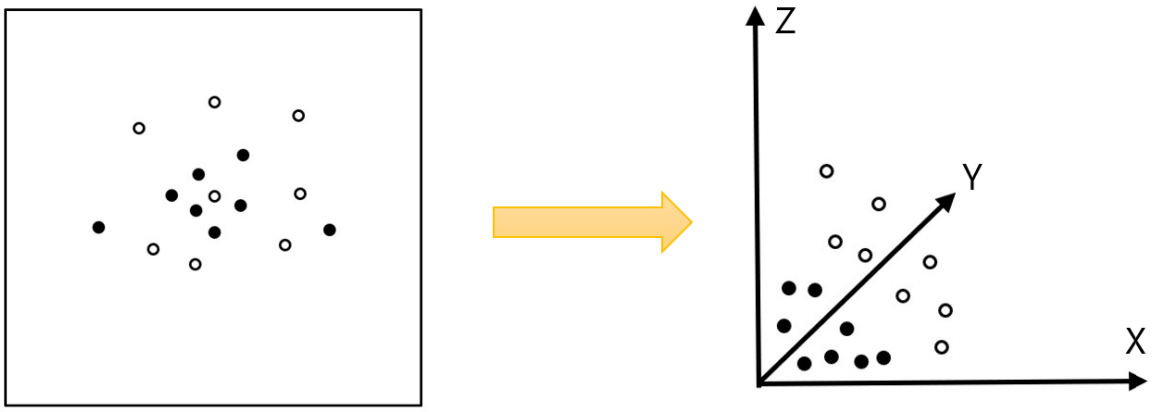


Figura 2.8: Transformación de datos del espacio original R_N al espacio de características H.

En la práctica, los tres tipos de kernel SVM más utilizados son el lineal (SMV$_l$), el gaussiano (SMV$_g$) y el polinomial (SMV$_p$), siendo los tipos de SVM que se estudian en este trabajo junto con la SVM sin kernel.

Lineal Si se emplea un Kernel lineal, la transformación en el espacio H es equivalente a la representación original en R_N. La ecuación (2.32) define matemáticamente la función Kernel lineal, la cual viene

representada en la Figura 2.9.

$$k(\vec{x},\vec{z}) =< \vec{x}\cdot\vec{z} >= \sum_{i=1}^{l} \vec{x}_i\vec{z}_i \tag{2.32}$$

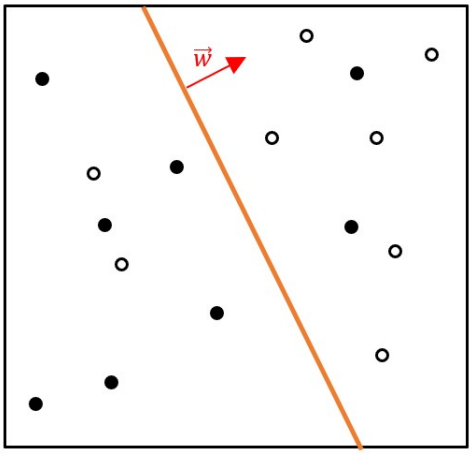


Figura 2.9: SVM con kernel lineal.

Polinomial La transformación Kernel polinomial permite determinar un hiperplano (línea continua naranja) que no sea una línea recta, es decir, está definido por un polinomio de grado r que puede tomar la forma según su ecuación. La ecuación (2.33) define matemáticamente la función Kernel polinómica, la cual viene representada en la Figura 2.10.

$$k(\vec{x},\vec{z}) = (< \vec{x}\cdot\vec{z} > +c)^r \tag{2.33}$$

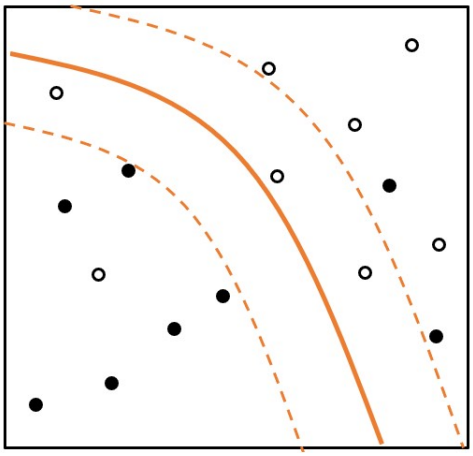


Figura 2.10: SVM con kernel polinómica.

Gaussiana Se hace uso de un Kernel gaussiano cuando los datos tienen una distribución circular. Es por ello que los hiperplanos (línea continua naranja) usados en este tipo de Kernel son circulares. La ecuación (2.34) define matemáticamente la función Kernel gaussiana, la cual viene representada en la Figura 2.11.

$$k(\vec{x},\vec{z}) = e^{\frac{||\vec{x}-\vec{z}||^2}{2\sigma^2}} \tag{2.34}$$

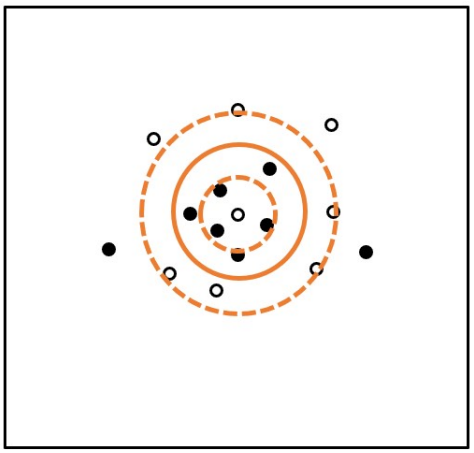


Figura 2.11: SVM con kernel gaussiana.

Debido a que las distribuciones de los datos de los ejercicios que se van a tratar no se conocen, se hace uso de los algoritmos de SVM y de sus respectivas kernels, para así determinar cuál de ellas es la que más se aproxima a la distribución de los datos en la realidad y que pueda ofrecer los resultados más precisos acorde a sus distribuciones.

2.2.3 Random Forests (RF)

RF es un algoritmo de Machine Learning que se puede usar para la clasificación de datos y se compone de una gran cantidad de pequeños árboles de decisión, llamados estimadores, cada uno de los cuales produce sus propias predicciones. Este algoritmo combina las predicciones de los estimadores para producir una predicción más exacta y fiable y funciona muy bien con datos de gran tamaño.

Debido a que son algoritmos extremadamente robustos y óptimos para tipos de datos heterogéneos, los RF son, a menudo, el punto de inicio al desarrollar un nuevo sistema de Machine Learning, ya que permite obtener una descripción general y rápida de qué tipo de precisión se puede lograr razonablemente en un problema.

La finalidad de los RF es encontrar una función predictora $f(X)$ capaz de predecir Y, que representa la respuesta de valor real. La función predictora está determinada por una función de pérdidas que está definida para minimizar el valor esperado del error.

Esta función de pérdidas $L(Y, f(X))$ es una medida que indica lo cerca que está $f(X)$ de Y y penaliza el valor de $f(X)$ que está muy alejado de Y. Se denomina pérdida *zero-one* y viene definida en la ecuación (2.35).

$$L(Y, f(X)) = \begin{cases} 0 & si \quad Y = f(X) \\ 1 & otros \end{cases} \tag{2.35}$$

En los métodos de clasificación, para determinar el valor de la función predictora se minimiza el error de las pérdidas *zero-one*. Esta función predictora, definida en la ecuación (2.36), se determina mediante la clase más frecuentemente predicha entre todos los estimadores.

$$f(x) = \max_{y \in Y} \sum_{j=1}^{J} I(h_j(x) = y) \tag{2.36}$$

Siendo $h_j(x)$ el conjunto de estimadores y $j = (1, ..., J)$ el número de estimadores dentro del algoritmo.

Para explicar los RF se detalla, en primer lugar, cómo se construyen los árboles de decisión. Los árboles de decisión son modelos que permiten la visualización instantánea del proceso de toma de decisiones. Su principal limitación es que un único árbol de decisión se adapta muy fuertemente a su conjunto de entrenamiento y se generaliza mal a ejemplos invisibles para el clasificador.

El proceso básico para el desarrollo del algoritmo de RF viene ilustrado en la Figura 2.12 cuyos pasos están expuestos a continuación:

1. Toma de muestras: Dado un conjunto de entrenamiento de N ejemplos, se toman muestras, repetidamente, de subconjuntos de datos de entrenamiento de tamaño n (donde $n < N$). Este submuestreo de un conjunto de entrenamiento se denomina agregación *bootstrap*, o empaquetado.

2. Método del subespacio aleatorio: Si cada ejemplo de entrenamiento tiene M características, se toma un subconjunto de ellas de tamaño $w < W$ para entrenar cada estimador. Por lo tanto, ningún estimador ve el conjunto de entrenamiento completo, cada estimador solo ve w características de n ejemplos de entrenamiento.

3. Estimadores de formación: Se crean árboles de decisión Ntree, o estimadores, y se entrenan a cada uno en un conjunto diferente de w características y n ejemplos de entrenamiento.

4. Predicción del resultado final: Para hacer una predicción de un nuevo ejemplo entrante, se pasan las características relevantes de este ejemplo a cada uno de los estimadores Ntree. Se obtendrán predicciones de Ntree, que deben combinarse para producir la predicción general del bosque aleatorio. En el caso de la clasificación, se usa la "votación por mayoría" para decidir la clase predicha.

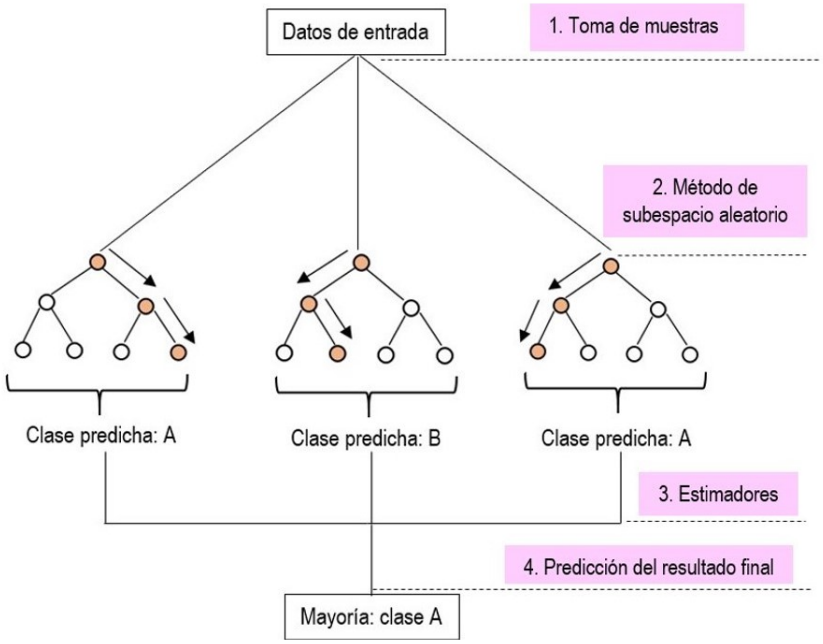


Figura 2.12: Diagrama de funcionamiento del algoritmo de clasificación de RF.

Este uso de muchos estimadores es la razón por la que el algoritmo de bosque aleatorio se denomina método de conjunto. Cada estimador individual es débil, pero cuando se combinan muchos estimadores

débiles, pueden producir un modelo mucho más fuerte. Los métodos de conjunto adoptan un enfoque denominado "fuerza en números", donde la salida de muchos modelos pequeños se combina para producir una predicción mucho más precisa y exacta.

La baja correlación entre modelos es clave para producir predicciones de conjunto más precisas que las predicciones individuales. La razón es que los árboles se protegen unos a otros de sus errores individuales. Si bien algunos árboles pueden estar equivocados, muchos otros árboles estarán en la decisión correcta, por lo que, como grupo, los árboles pueden moverse en la dirección correcta. Entonces, uno de los requisitos previos para que el RF funcione bien es que las predicciones, y por lo tanto los errores, realizados por los árboles individuales deben tener correlaciones bajas entre sí.

Cuando se toma una muestra de bootstrap de los datos, algunas observaciones no se encuentran dentro de esa muestra, es decir, se encuentran excluidas de ella. Estas muestras se denominan datos fuera de la bolsa u out-of-bag (OOB) y se utilizan para obtener una estimación del error de clasificación a medida que se agregan árboles al bosque. Este error se estima internamente durante la ejecución. Para poder estimar el error de clasificación, se define primero la predicción de los datos fuera de la bolsa (2.37).

$$\hat{f}_{oob}(x_i) = \max_y \sum_{j \in J} I(\hat{h}_j(x_i) = y) \tag{2.37}$$

Para clasificaciones con pérdidas *zero-one*, la estimación del error de clasificación, haciendo uso de la predicción de los datos fuera de la bolsa de (2.37), es la definida en la ecuación (2.38).

$$E_{oob} = \frac{1}{N} \sum_{i=1}^{N} I(y_i \neq \hat{f}_{oob}(x_i)) \tag{2.38}$$

El algoritmo de RF funciona muy bien para problemas de clasificación de datos. Se mantiene estable ante nuevas entradas, ya que el valor final sigue siendo el promedio de todos los valores predichos por los estimadores. Es debido a esto que se hace uso de dicho algoritmo a la hora de clasificar los ejercicios, ya que se trata de un algoritmo robusto y cuyas decisiones se deciden mediante el consenso de muchos predictores lo que da lugar a una clasificación más detallada y precisa. Sin embargo, este algoritmo es mucho más complicado de implementar que un solo árbol de decisión y, además, puede requerir un gran tiempo de entrenamiento. Por último, es muy difícil de interpretar y comprender el funcionamiento de todos los árboles creados dentro del algoritmo.

2.2.4 K-Nearest Neighbours (KNN)

El algoritmo KNN es uno de los algoritmos de clasificación más simples y con alta precisión. A pesar de su simplicidad, KNN puede superar a los clasificadores más potentes y se usa en una variedad de aplicaciones como la medicina, las plataformas de contenido digital, los procesos de venta cruzada o los sistemas de recomendación, entre otras muchas.

KNN es un algoritmo de aprendizaje supervisado, es decir, que a partir de un conjunto de datos inicial su objetivo será el de clasificar correctamente todos los datos nuevos. El conjunto de datos típico de este tipo de algoritmos está formado por varias características y una clase. El algoritmo clasifica cada dato nuevo en el grupo que corresponda, según tenga K vecinos más cerca de un grupo o de otro. Es decir, calcula la

distancia del elemento nuevo a cada uno de los existentes, y ordena dichas distancias de menor a mayor para ir seleccionando el grupo al que pertenecen. Este grupo será, por tanto, el de mayor frecuencia con menores distancias.

En la Figura 2.13 se puede observar un ejemplo en el que hay dos clases y se tiene que identificar a qué clase pertenece el nuevo dato introducido (círculo verde).

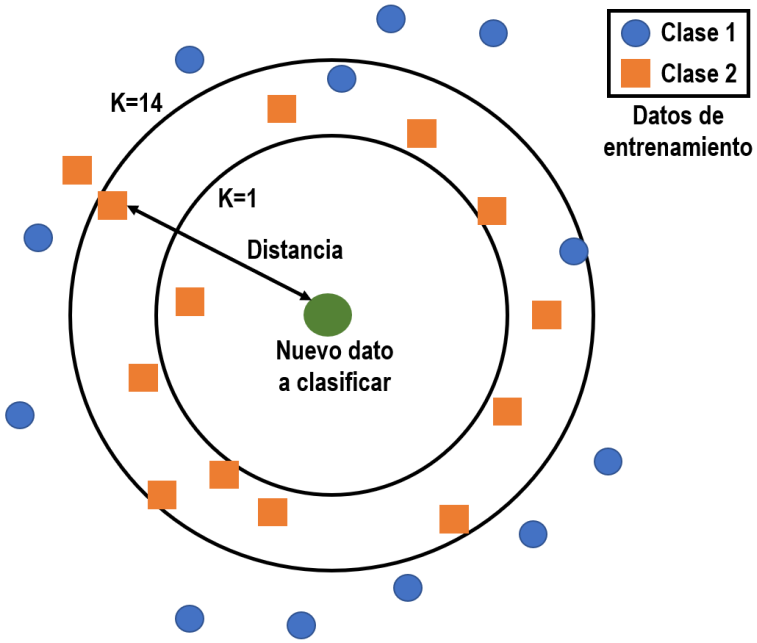


Figura 2.13: Esquema del funcionamiento del algoritmo de KNN.

La asignación de la clase de un nuevo dato depende del número K, es por ello que el algoritmo lo primero que hace es escoger el valor K y su distancia. Después, encuentra el K vecino más cercano del conjunto de datos que se quiere clasificar. Y, por último, se asigna la etiqueta de clase por "votación mayoritaria".

Para poder determinar dicha distancia, el algoritmo se basa en la distancia euclidiana entre una muestra de prueba y las muestras de entrenamiento especificadas. Sea x_i una muestra de entrada con p características $(x_{i1}, x_{i2}, ..., x_{ip})$, h el número total de muestras de entrada $(i = 1, 2, ..., h)$ y p el número total de características $(j = 1, 2, ..., p)$. La distancia euclidiana entre la muestra x_i y x_l $(l = 1, 2, ..., h)$ se define en la ecuación (2.39).

$$d(x_i, x_l) = \sqrt{(x_{i1} - x_{l1})^2 + (x_{i2} - x_{l2})^2 + ... + (x_{ip} - x_{lp})^2} \tag{2.39}$$

La ventaja de hacer uso del algoritmo de KNN es que no se necesita hacer ninguna suposición sobre los datos a clasificar. Se pueden clasificar datos complejos haciendo uso de funciones sencillas como aproximaciones locales. Además se trata de un algoritmo simple en el que cada dato de prueba es comparado contra todo el conjunto de datos de entrenamiento, es decir, no determina una función de clasificación como los algoritmos anteriores. A este tipo de algoritmos se les llama *lazy learning methods*. Además, se trata de un algoritmo muy sensible a la variable K, de modo que con valores distintos de K se puede obtener resultados muy distintos. Este valor suele fijarse tras un proceso de pruebas con varias clases.

Será la bondad de los resultados lo que determinará el ajuste de aspectos del algoritmo como el propio valor K. Es por ello, que se hace uso de KNN en este proyecto ya que ofrece una muy buena precisión ante datos con multitud de clases.

Capítulo 3

Desarrollo del algoritmo propuesto

El objetivo principal de este proyecto es evaluar e identificar ejercicios empleando algoritmos de Machine Learning. Para dicha evaluación y detección se realizan dos tipos de clasificaciones en las cuales se utiliza la información de la velocidad angular y aceleración lineal, incluyendo únicamente en la segunda de ellas los ángulos de movimiento experimentados por las IMUs. El fin de introducir la información extra de los ángulos es ver si los resultados obtenidos al clasificar e identificar los ejercicios mejoran o empeoran. Por último, se estudia la influencia de la posición de las IMUs y las posibles combinaciones existentes de las posiciones sobre las extremidades corporales.

En este apartado del proyecto se explican de manera detallada tanto los algoritmos estudiados para clasificar los ejercicios como el procesado de las señales para la evaluación de los mismos. Para ello, se definen los recursos y materiales utilizados para obtener las señales del movimiento y se determinan los diferentes algoritmos y tipos de clasificación utilizados.

3.1 Ejercicios analizados

Para evaluar la ejecución de ejercicios pautados es fundamental definir la forma correcta de realizarlos. Asimismo, es importante definir el proceso de seguimiento y evaluación de los mismos, que incluye la definición de sensores a utilizar, los pasos a seguir para de grabar dichos ejercicios y la sincronización de la información obtenida.

Los voluntarios han realizado un número determinado de ejercicios, en concreto siete ejercicios diferentes, elegidos basándose en las ruedas de de ejercicios propuestas en el proyecto de Vivifrail [11], en el que se asignan determinadas rutinas físicas a los pacientes según su estado funcional.

Estos ejercicios, que se pueden observar en la Figura 3.1, son:

1. Flexo-extensión de la pierna (fes): se parte de la posición de aproximadamente 90° de flexión de rodilla y se lleva hasta la extensión de la misma, aproximadamente 0°. Se graban ambas piernas de forma independiente, resultando en fes de la pierna derecha (fesR) y de la izquierda (fesL).

2. Sentadillas (sen): partiendo de la posición erguida, el voluntario se aproxima a la posición de sentarse

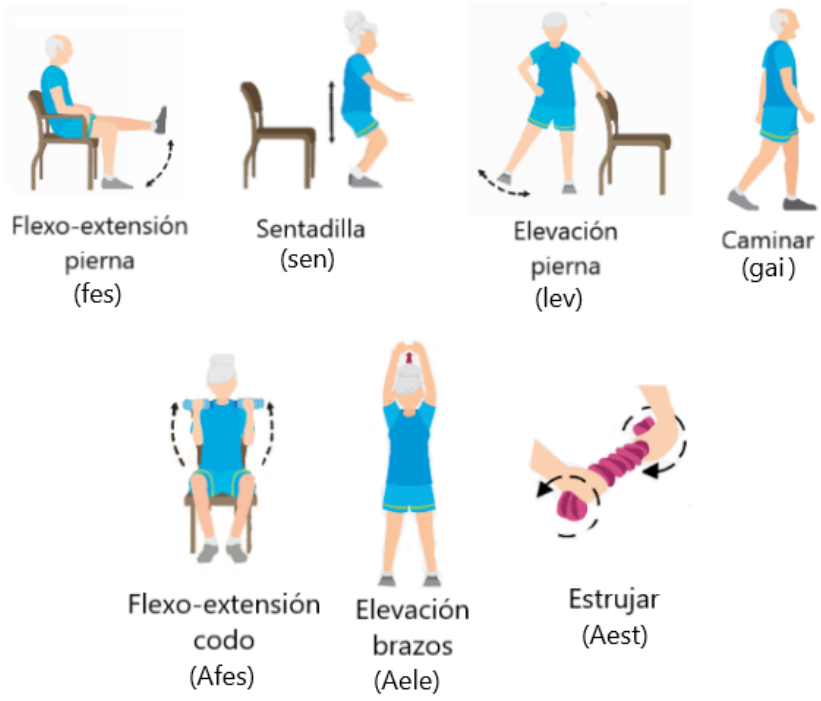


Figura 3.1: Ejercicios elegidos de la rueda de Vivifrail [11].

hasta que toca ligeramente la silla y en ese momento vuelve de nuevo a posición erguida.

3. Elevación lateral de pierna (lev): manteniéndose de pie, el voluntario eleva la pierna lateralmente manteniéndola completamente extendida sin flexionar. Se graban ambas piernas de forma independiente, resultando en lev de la pierna derecha (levR) y de la izquierda (levL).

4. Caminar (gai): este ejercicio consiste en que el sujeto camina de forma libre por el espacio disponible.

5. Flexo-extensión de codos (Afes): se parte de la posición de extensión de brazos a lo largo del cuerpo y se flexionan los codos de forma simétrica hasta la posición de, aproximadamente, 90 º.

6. Elevación de los brazos por encima de la cabeza (Aele): con las manos entrelazadas, se elevan los brazos por encima de la cabeza, realizando un giro de unos 180 º con el hombro, manteniendo los brazos estirados.

7. Estrujar (Aest): sujetando con ambas manos un tejido, se realiza el movimiento de "estrujar", en el que se giran de forma antisimétrica las muñecas.

El número de repeticiones de cada ejercicio, así como el número de series, se determinó para tener el máximo número de repeticiones de cada individuo sin que se fatigue. Se realizaron para todos los ejercicios una serie de 20 repeticiones, excepto para el ejercicio de sentadillas, que se hicieron 15. Se realizaron un total de cuatro series con el número de repeticiones correspondiente de cada ejercicio: las dos primeras se hicieron de una forma correcta, realizando los movimientos acorde con la descripción dada previamente, mientras que las dos últimas se hicieron incorrectamente, desviando los movimientos a la hora de realizar los ejercicios.

3.2 Sistemas sensoriales

En este proyecto se han usado cuatro IMUs para grabar los ejercicios elegidos. En concreto, se han empleado las señales de aceleración lineal y velocidad angular recogidas por los sensores inerciales durante la ejecución de los movimientos.

3.2.1 Sensores inerciales

Las IMUs utilizadas en el proyecto son las NGIMUs de x-io Technologies [12], las cuales combinan sensores y algoritmos de procesamiento de datos para crear una plataforma adecuada de aplicaciones de registro de datos y en tiempo real.

La IMU empleada es una plataforma multisensorial que dispone de giróscopos, acelerómetros y magnetómetros triaxiales, además de sensor de temperatura y humedad. En el caso del proyecto sólo se va a hacer uso de los giróscopos y acelerómetros que van a dar información de la velocidad angular y la aceleración lineal, respectivamente. La Figura 3.2 muestra el rango dinámico y precisión proporcionado por el fabricante. Otra de las características es que dispone de una tarjeta micro SD que permite almacenar los datos para procesarlos posteriormente *offline*.

Sensors		
Gyroscope	Range: Sample Rate: Resolution:	2000°/s 400 Hz 16-bit
Accelerometer	Range: Sample Rate: Resolution:	16 g 400 Hz 16-bit

On-Board Logging	
Memory	Micro SD Card
USB Download	Yes

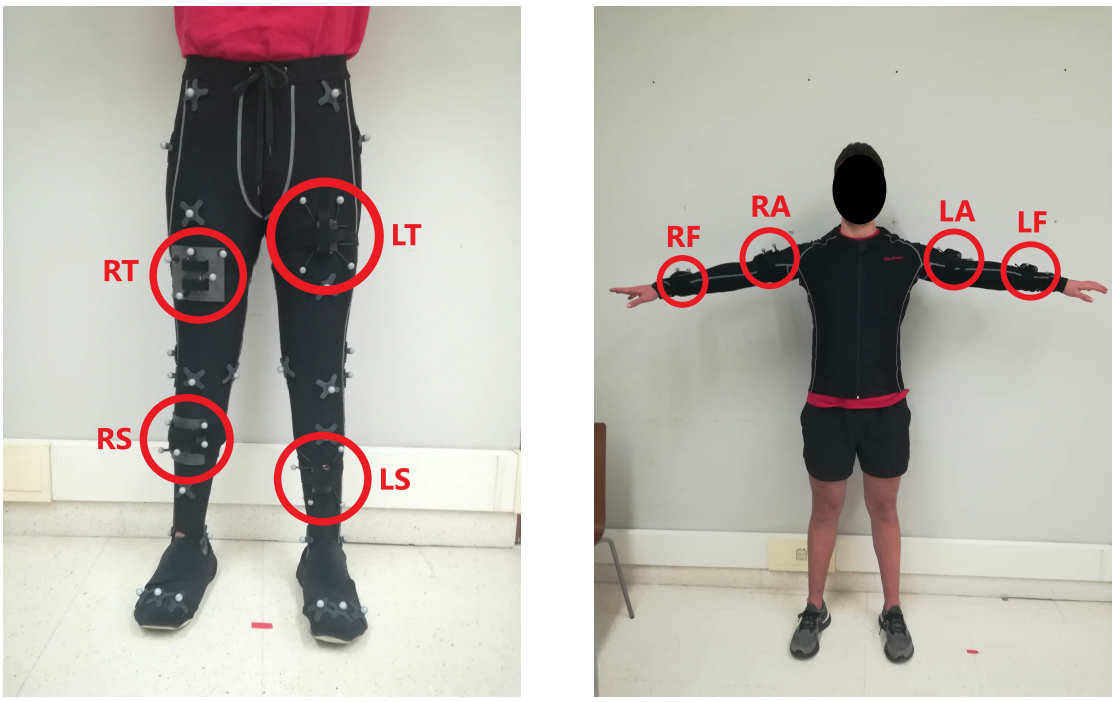
Figura 3.2: Margen, rango dinámico y precisión de las NGIMUs [12]

Como se puede observar en la Figura 3.2, la resolución de ambos sensores es de 16 bits y el rango de muestreo es de 400 Hz. A la hora de utilizar las IMUs, se configuraron con una frecuencia de muestreo de 100 Hz, es decir, que cada segundo se toman 100 muestras de la información del sensor.

3.2.2 Posición de las IMUs sobre el cuerpo

En este estudio se ha hecho uso de cuatro IMUs, las cuales se han usado tanto para los ejercicios de pierna como para los de brazo. De tal forma que, primero, se graban los ejercicios de pierna y después los de brazos.

Las IMUs se han colocado en ambos muslos y ambas espinillas, siguiendo las posiciones detalladas en [3], en el caso de los ejercicios de miembros inferiores, y en ambos brazos y antebrazos en el caso de ejercicios de miembros superiores, como se muestra en la Figura 3.3.



(a) Posición de las cuatro IMUs sobre las piernas.

(b) Posición de las cuatro IMUs sobre los brazos.

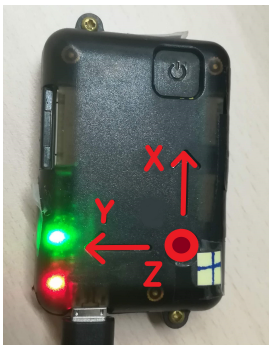
Figura 3.3: Posición de las cuatro IMUs sobre los brazos y las piernas.

En la Figura 3.3a se puede ver que las IMUs están nombradas como RT, RS, LT y LS, estas siglas indican si la IMU está posicionada en la extremidad derecha (R, de *right*) o en la izquierda (L, de *left*). Además, las posiciones de las IMUs sobre las piernas se diferencian entre muslo (T, de *thigh*) y espinilla (S, de *shin*). Por su lado, las posiciones en los miembros superiores se distinguen si son en el brazo (A, de *arm*) o en el antebrazo (F, de *forearm*). Se han elegido estas posiciones por la sencillez para colocar los sensores.

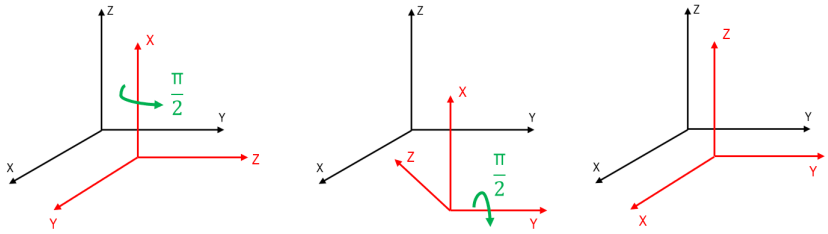
Todas las IMUs que se usan para la detección de los ejercicios están colocadas de la misma forma sobre las personas. Estas IMUs se colocan en el cuerpo con la orientación que se muestra en la Figura 3.4a, es decir, con el eje X apuntando hacia el techo cuando el voluntario está erguido.

Los cambios de sistema de referencia son muy importantes para poder determinar correctamente los ángulos que determinan la orientación de las IMUs al realizar los ejercicios y poder clasificar correctamente sus características. Los ángulos se determinan respecto del sistema de referencia global, definido en la Figura 3.4b.

Como se puede observar en la Figura 3.4b, el sistema de referencia de la IMU (en color rojo) no es el mismo que el sistema de coordenadas global (en color negro), por lo que se realiza un cambio de sistema de referencia. Para hacer el cambio y obtener los ángulos equivalentes en el sistema de referencia global se hacen dos giros. El primero de ellos es un giro de $\pi/2\,\text{rad}$ o de $90\,^{\text{o}}$ sobre el eje X y cuando este giro se ha producido, se realiza el otro giro de $\pi/2\,\text{rad}$ o $90\,^{\text{o}}$ sobre el eje Y, como se muestra en la Figura 3.4b. Tras este cambio de sistema de referencia, se obtienen los ángulos Roll y Pitch tal y como se explica en



(a) Sistema de referencia de la IMU.



(b) Giros para la transformación del sistema de referencia de la IMU al sistema de referencia global. Siendo el sistema de referencia global el negro, y el sistema de referencia de las IMUs el rojo.

Figura 3.4: IMU en su orientación sobre el cuerpo (eje X vertical hacia el techo) y su sistema de referencia junto con el sistema de referencia global.

el apartado 2.1.2 del estado del arte.

3.2.3 Sincronización de los datos obtenidos por las IMUs

Las IMUs, individualmente, no comparten información de conexión entre ellas y hace que las señales que generan cada una no estén sincronizadas, por lo que es muy importante determinar un modo de sincronización. En este trabajo se realizan unos movimientos de sincronización que se ejecutan con anterioridad a cada ejercicio. Así, detectando eventos significativos, como el cruce por cero de la velocidad, presentes en las señales de varios sensores que se mueven de manera simultánea, se puede realizar la sincronización de sus señales

Los movimientos de sincronización para los ejercicios de pierna se pueden observar en la Figura 3.5 y son los siguientes:

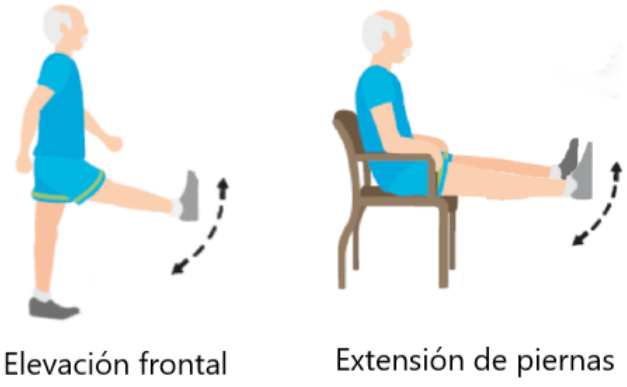


Figura 3.5: Movimientos para la sincronización de las piernas.

1. Elevación frontal de la pierna derecha con la rodilla estirada.
2. Elevación frontal de la pierna izquierda con la rodilla estirada.

3. Flexo-extensión de rodilla con ambas piernas a la vez, juntando los pies para asegurar el sincronismo del movimiento.

Se realizan dos repeticiones de los movimientos de sincronización. El movimiento de sincronización de la elevación frontal de cada pierna sirve para sincronizar las dos IMUs que se encuentran en el muslo y espinilla de la misma. El movimiento de flexo-extensión de rodilla con ambas piernas se realiza para poder sincronizar ambas piernas. Como las dos IMUs de la pierna derecha están sincronizadas entre sí, igual que las de la pierna izquierda, para poder sincronizar las cuatro IMUs, es suficiente con mover una parte de las piernas, en este caso la espinilla.

En los ejercicios de brazo solamente se hace un movimiento ya que es muy fácil mover ambos brazos a la vez, lo que da lugar a la posibilidad de sincronizar las cuatro IMUs situadas en los brazos simultáneamente. Este movimiento consiste en la elevación frontal de los dos brazos, con las manos entrelazadas mirando hacia arriba y con los codos bloqueados, como se detalla en la Figura 3.6

Figura 3.6: Movimientos para la sincronización de los brazos.

Estos movimientos de sincronización se ven reflejados en las señales empleadas para sincronizar manualmente los diferentes ejercicios. En la Figura 3.7 se representa el valor absoluto de la velocidad en cada una de las cuatro IMUs en un ejercicio de flexo-extensión de la pierna derecha. La primera fila representa la IMU situada en la espinilla derecha (RS), la segunda fila representa a la IMU situada en el muslo derecho (RT), la tercera fila representa a la IMU situada en la espinilla izquierda (LS) y la última fila representa la IMU situada en el muslo derecho (LT).

En la Figura 3.7, los cuatro picos rodeados por los círculos verdes representan al movimiento de sincronización de la pierna derecha, los cuatro picos rodeados por los círculos rojos representan al movimiento de sincronización de la pierna izquierda y los cuatro picos rodeados por los círculos azules representan al movimiento de sincronización de ambas piernas. Los puntos rojos remarcados en el medio de los cuatro picos en la Figura 3.7 son los que se eligen para determinar la sincronización de la pierna derecha, de la pierna izquierda y de ambas piernas dependiendo en el círculo en el que se encuentren.

Por otro lado, se define el intervalo de interés para el análisis como el tiempo en el que se está realizando el ejercicio a estudiar. A la hora de determinar el tiempo de inicio y fin del ejercicio, se descartan la

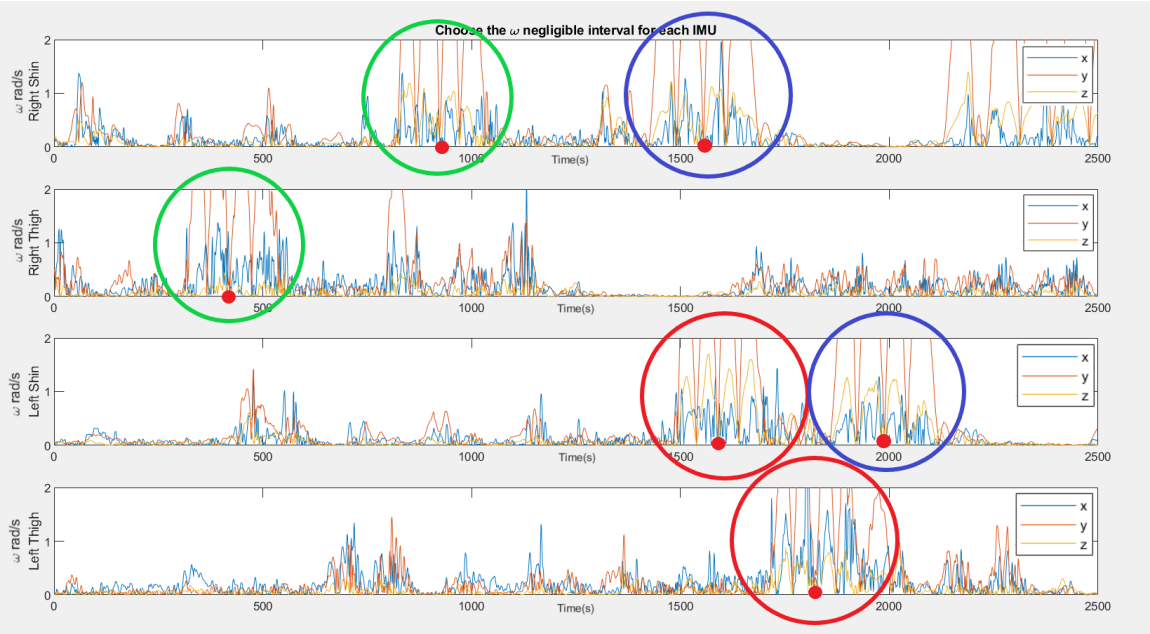


Figura 3.7: Señales correspondientes a los movimientos de calibración. Los colores indican los movimientos iguales dos a dos para cada una de estas señales. Además, se muestra la elección del tiempo de sincronización de las dos IMUs de la pierna derecha, de las dos IMUs de la pierna izquierda y de las cuatro IMUs de ambas piernas.

primera y última repetición del mismo ya que pueden ser erróneas. Por lo tanto, el tiempo de inicio se toma como el inicio del tercer pico de la velocidad y el tiempo final como el fin del antepenúltimo pico de la velocidad. En la Figura 3.8 se muestra el valor absoluto de la velocidad angular del ejercicio de flexo-extensión de la pierna derecha junto con los puntos de inicio y fin seleccionados del ejercicio.

En la Figura 3.9 se muestra un ejemplo de las señales ya sincronizadas para el ejercicio de flexo-extensión de la pierna derecha bien realizado. En el momento en el que la sincronización y los tiempos de inicio y fin del ejercicio están determinados, las señales de dicho ejercicio están preparadas para ser segmentadas y para poder sacar características de las mismas.

3.2.4 Pasos para la grabación de los ejercicios

El orden en la ejecución de los ejercicios es muy importante y conviene que sea igual en todos ellos, ya que si no, la información grabada será más difícil de localizar y etiquetar de forma correcta, pudiendo dar lugar a errores a la hora de clasificar.

El proceso de grabación en este trabajo ha sido el siguiente: primero encender todas las IMUs, para a continuación realizar los movimientos de sincronización de las mismas. Tras ello, se realiza una parada de unos pocos segundos, de forma que el intervalo de señal del ejercicio quede claramente separado del correspondiente a los movimientos de calibración, y posteriormente se empieza a hacer el ejercicio determinado. Por último, cuando se ha terminado de hacer las repeticiones se apagan todas las IMUs.

Este procedimiento se realiza en cada grabación. En este trabajo se realiza una grabación por ejercicio, por lo que se repite tantas veces como ejercicios se han grabado.

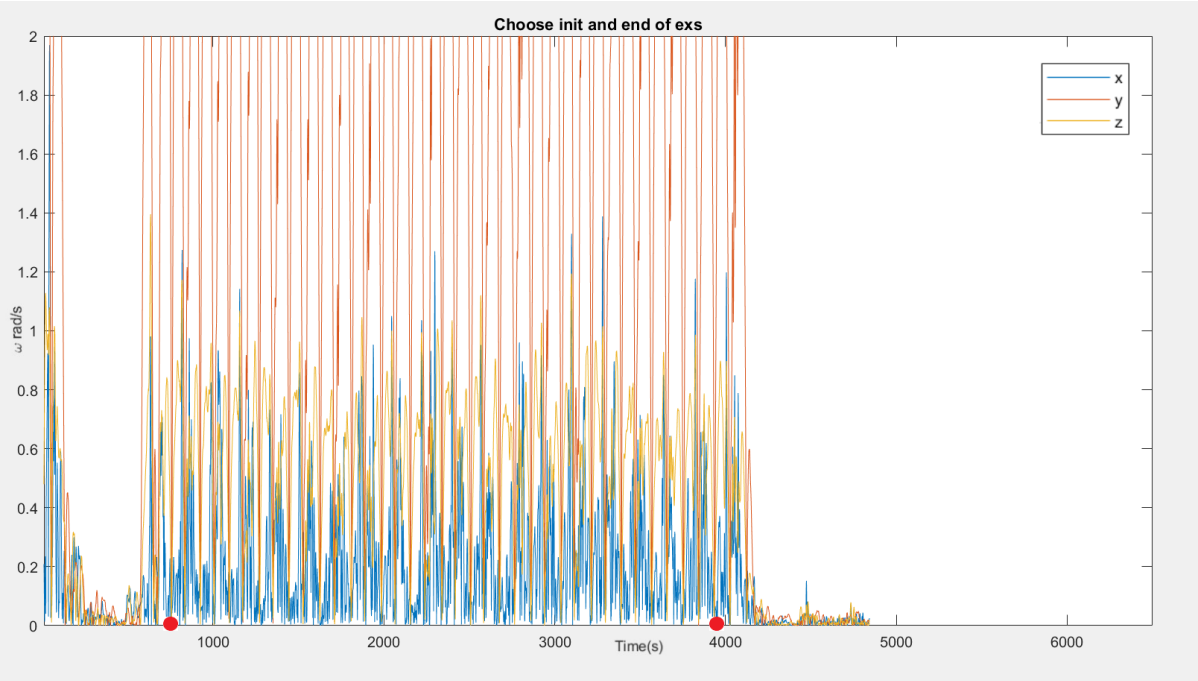


Figura 3.8: Señal de la velocidad angular en valor absoluto medida por la IMU RS durante el ejercicio de fesR. Los diferentes colores distinguen las componentes X (azul), Y (rojo) y Z (amarillo) de la señal medida por el giróscopo de la IMU.

3.2.5 Volcado y organización de datos

La organización y el volcado de datos es importante ya que si se cometen errores en el renombramiento de los archivos, dará lugar a un aumento en el número de errores de la clasificación, debido a que habrá ejercicios llamados erróneamente como otros y sus características serán diferentes.

Para el volcado de datos, se conectaban las IMUs al ordenador vía USB y se copian todos los archivos "*.*XIO*" generados por las IMUs. Después, se transforman esos archivos con el *software* propio de NGIMU, *NGIMU SD Card File Converter* [12], en archivos "*.*csv*", y, por último, se renombran siguiendo un formato estandarizado para la identificación de los datos.

Tal y como se muestra en el Anexo A, un único archivo "*.*XIO*" da lugar a varios archivos que incluyen información de los diferentes sensores que contienen las IMUs, como la información del movimiento, temperatura o humedad relativa. En este proyecto sólo se hace uso del archivo denominado "*sensors.csv*" ya que es el que contiene los datos del movimiento, entendidos como las velocidades angulares y las aceleraciones lineales de los sensores, que son los que empleamos a lo largo de este trabajo.

3.3 Base de datos

La toma de datos se realizó con voluntarios que firmaron previamente un consentimiento informado. El proyecto ha sido evaluado y aprobado por el Hospital Universitario de Guadalajara (Junta de Revisión Institucional No.2018.22.PR, versión V.1. con fecha 21/12/2020) en el marco del proyecto Frailcheck (SBPLY/17/180501/000392) y siguiendo las recomendaciones relacionadas con la COVID-19.

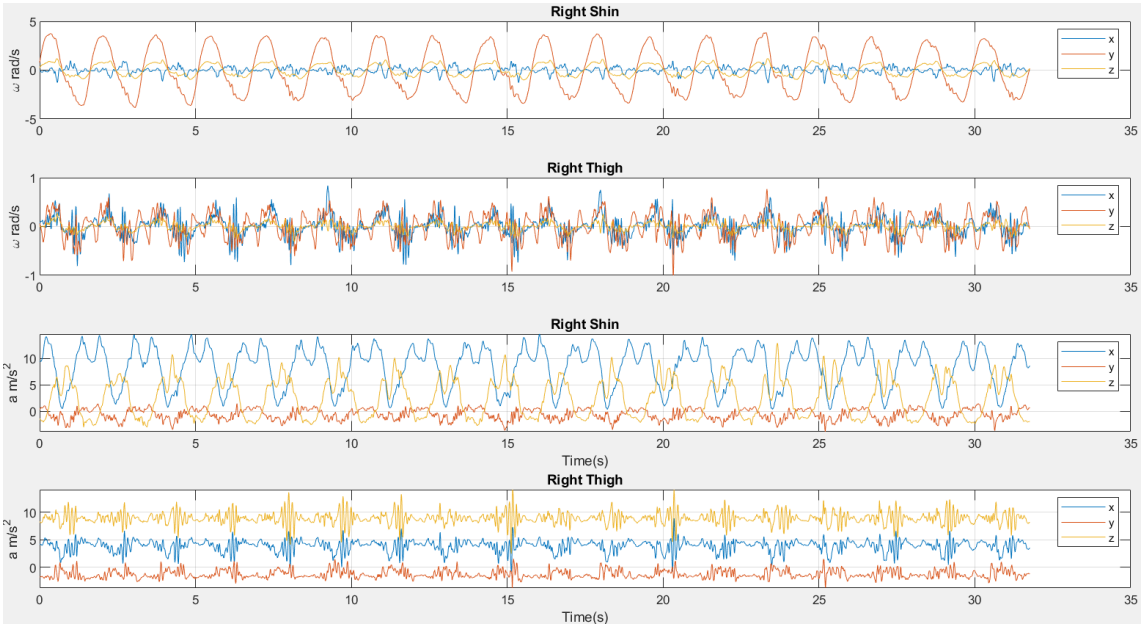


Figura 3.9: Ejercicio sincronizado y recortado de la pierna derecha. En la primera y la tercera gráfica se representa el valor de la velocidad angular y la aceleración lineal de la espinilla derecha, respectivamente. Y en la la segunda y la cuarta gráfica se representa el valor de la velocidad angular y la aceleración lineal de el muslo derecho, respectivamente.

Toda la información recogida a través de la grabación de los ejercicios se organiza en una base de datos generada en Excel, cuya muestra consta de 14 voluntarios. De esos voluntarios, 4 de ellos son mujeres y 10 hombres. La edad media de los voluntarios es de 31 ± 11 años, su altura es de 172 ± 6 cm y su peso es $70{,}7 \pm 8{,}9$ kg.

La información de los voluntarios se agrupa en una base de datos para evitar posibles errores de clasificación, que contiene información relevante sobre la realización de los ejercicios. Todos aquellos ficheros que se sabían que habían tenido problemas durante la grabación no se incluyeron en la base de datos.

En la Figura 3.10, se puede observar la forma de introducir la información al registro de datos de un voluntario etiquetado como "Voluntario x". En la primera fila se determina el identificador del voluntario y los ejercicios que realiza. En la segunda fila se determina la fecha en la que ese voluntario ha realizado los ejercicios y, además, se señala con una x cada ejercicio realizado. Las siguientes cuatro filas contienen un número el cual indica el nombre de los archivos que guardan las IMUs al grabar. Y la última fila contiene el número de repeticiones que se ha realizado en cada ejercicio y los comentarios relevantes sobre dicho ejercicio.

La persona responsable de la grabación de cada día fue quien determinaba si el ejercicio era correcto, si la sincronización estaba bien hecha y si las IMUs estaban bien encendidas o no a la hora de grabar. Por lo que el *ground truth* se ha determinado a partir de la observación de dicha persona, quien tenía que tener pleno conocimiento tanto de los ejercicios como de la sincronización y del funcionamiento de las IMUs.

Voluntario x	Etiqueta	RCalib	LCalib	fesR0_1	fesR0_2	fesR1_1	fesR1_2	fesL0_1	fesL0_2
20201212	x si ejercicio hecho	x	x	x	x	x	x	x	x
#IMUsL	Nº de archivo	498	499	500	501	502	503	504	505
#IMUtL	Nº de archivo	956	957	958	959	960	961	962	963
#IMUsR	Nº de archivo	602	603	604	605	606	607	608	609
#IMUtR	Nº de archivo	428	429	430	431	432	433	434	435
Notas	Nº de repeticiones hechas y anotaciones de incidencias			20	20- Mueve la pierna izquierda en vez de la derecha. DESCARTADA	20	20	20	20

Figura 3.10: Ejemplo de recopilación de información y creación de la base de datos.

3.4 Análisis de datos

Tras tener las señales de interés se determinan las características necesarias a la hora de clasificar dichos ejercicios y el tipo de segmentación a usar.

3.4.1 Características elegidas

A la hora de clasificar los ejercicios con algoritmos de Machine Learning, se han empleado cuatro características empleadas con frecuencia en la literatura, como en [8, 5, 13]. Estas son: el máximo, el mínimo, la media y la desviación estándar de la señal a estudiar. Las cuales se definen como:

- El máximo y el mínimo de una señal son los valores más grandes y más pequeños, respectivamente, que toma una señal, ya sea en una región o en todo su dominio.

- La media $\bar{x}$ es el valor promedio de un conjunto de datos numéricos, calculada como la suma del conjunto de valores dividida entre el número total de valores. Su expresión viene definida en la ecuación (3.1).

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} \tag{3.1}$$

  donde N es el número de muestras. Siendo x, en el caso del proyecto, cada componente de la velocidad angular, la aceleración lineal o ángulo de giro.

- Por último, la desviación estándar σ_x es una medida que ofrece información sobre la dispersión media de una señal. La desviación es la separación que existe entre un valor cualquiera de la señal x_i y la media $\bar{x}$. Su expresión viene definida en la ecuación (3.2)

$$\sigma_x = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N}} \tag{3.2}$$

3.4.2 Segmentación de los datos

La segmentación es la forma de obtener diferentes intervalos relevantes de una señal de los que extraer características a partir de una señal completa. Es por ello que la elección de la segmentación tiene que ser correcta para que los resultados sean óptimos. Una elección subóptima tanto del tamaño de ventana como del tamaño de step puede dar lugar a un aumento de los errores a la hora de clasificar.

A la hora de elegir la segmentación óptima también se recurrió a la literatura de la temática del proyecto. En muchas de las propuestas se hace uso de una segmentación por ventana deslizante, como en la propuesta de E. Preatoni et al. [3], lo que justifica el que sea también la elección para la segmentación de los datos en este trabajo.

La segmentación por ventana deslizante se caracteriza por el tamaño de la ventana y el tamaño de step o salto de una ventana a otra. Otra característica típica de este tipo de segmentación es el solapamiento, que son los intervalos de señal que son comunes en dos ventanas consecutivas. El solapamiento, por tanto, viene determinado por el tamaño de ventana y del salto, como se presenta en la Tabla 3.1.

Tabla 3.1: Ejemplo de las muestras de señal seleccionadas en una segmentación por ventana deslizante.

1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12

En la Tabla 3.1, se muestra una señal de 12 muestras, el tamaño de la ventana deslizante (cuadrados coloreados) es 6 y el tamaño del step es 1. El tamaño del step es 1 ya que de un instante de tiempo a otro la ventana se ha movido una posición a la derecha. Todas estas características dan lugar a un solapamiento de la señal. En la primera iteración y en la segunda, se solapan las muestras de la 2 a la 6, y entre la segunda y la tercera iteración se solapan las muestras de la 3 a la 7. Es por ello, que en el paso de la segmentación por ventana deslizante es tan importante elegir bien las características que se van a obtener de los datos de los ejercicios, como el óptimo estudio y elección del tamaño de ventana y del tamaño de step.

En este trabajo el tamaño de ventana de un ejercicio se establece como el número de muestras que se necesita para realizar una repetición de dicho ejercicio utilizando una frecuencia de adquisición de 100 muestras/s. Por ejemplo, si en un ejercicio de caminar se tarda un segundo y medio en dar un paso, el tamaño de ventana será de 150 muestras.

Para obtener el tamaño de ventana óptimo de las repeticiones de los ejercicios segmentados, se calcula la media del tiempo que se tarda, para todos los voluntarios, en hacer cada una de las repeticiones de cada ejercicio. La media se hace sólo de los ejercicios bien hechos ya que se trata de los más representativos y, además, porque los mal hechos se realizan más rápido.

Los resultados obtenidos se muestran en la Tabla 3.2, donde se indica el tamaño de ventana y el tamaño de step óptimo para la clasificación binaria y para las clasificaciones diferentes de multiclase.

Para estimar la ventana en la clasificación binaria se toman los tiempos promedios para cada ejercicio, de forma que se optimiza para cada caso ya que no se tarda lo mismo en hacer todos los ejercicios. En

Tabla 3.2: Resultados del tamaño de ventana y del tamaño de step óptimos para la segmentación por ventana deslizante en los diferentes tipos de clasificación.

Ejercicio	**Tamaño de ventana**		**Tamaño de step**	
	Clas. binaria	**Clas. multiclase**	**Clas. binaria**	**Clas. multiclase**
fesRL	240	300	100	120
levRL	230			
sent	330			
gait	320			
Afes	260			
Aele	340			
Aest	240			

el caso de la clasificación multiclase se trabajan con todos los ejercicios simultáneamente por lo que es necesario tener una única ventana que sea incluya a todos los ejercicios. En ese caso se tomó el promedio redondeado a la alza de los tiempos de ejecución de los ejercicios cuyo redondeo es cercano a las ventanas de mayor longitud, se determina un tamaño de ventana para la clasificación multiclase de 300 muestras

Para la clasificación binaria se emplea un tamaño de ventana diferente para cada ejercicio ya que no lleva el mismo tiempo realizar todos los ejercicios. Los tamaños de ventana de la clasificación binaria son los que se pueden observar en la Tabla 3.2. En el caso de la clasificación multiclase, el tamaño de ventana para todos los archivos es el mismo porque se parte del supuesto de no saber qué ejercicio se está realizando y, por tanto, no se cuenta con la información del tamaño de ventana óptimo para cada uno de ellos.

El tamaño de step óptimo se determinó mediante un procedimiento de prueba-error en el que se evaluó la precisión de los clasificadores para distintos números de muestras. Se probó, para la clasificación binaria, con tamaños de steps de 40 muestras, los cuales daban una precisión extremadamente buena cuyo error de clasificación rondaba en torno al 0 %, aunque tenía el problema de que tardaba aproximadamente 40 minutos en ejecutarse cada iteración de la clasificación binaria de cada ejercicio. Después, se probó con un tamaño de step de 80 muestras, obteniendo una precisión que era prácticamente la misma que con un tamaño de step de 40 muestras, pero se tardaba 5 minutos menos en ejecutarse. A continuación se probó con un tamaño de step de 100 y de 110 muestras y se observó que, para el tamaño de step de 100 muestras, la precisión era bastante buena, es decir, el error de clasificación estaba en torno al $1\,\% - 2\,\%$, y el tiempo de ejecución se reducía considerablemente el cual era, aproximadamente, de unos 5 minutos para la clasificación de cada iteración. En cambio, para los valores del tamaño de step a partir de 110 muestras, la precisión disminuía conforme se iba aumentando el tamaño del step. Es por ello que para la clasificación binaria se impuso un tamaño de step de 100 muestras, cuyos errores de clasificación, dependiendo del clasificador, rondan el $1\,\% - 2\,\%$ y el tiempo de clasificación de cada ejercicio es de, aproximadamente, 5 minutos.

En el caso de la elección del tamaño de step óptimo para la clasificación multiclase, se llevó a cabo el mismo procedimiento que el realizado para la determinación del tamaño de step en la clasificación binaria. En este caso el tamaño de step que se obtuvo fue de 120 muestras, cuya precisión era menor al

1 % de error para cada clasificación multiclase, y el tiempo de ejecución se incrementaba a 40 minutos para cada iteración, el cual se consideró aceptable ya que se maneja mucha más cantidad de datos que en la clasificación binaria.

3.5 Clasificación

El objetivo principal de este proyecto es determinar el algoritmo que proporcione mejores resultados para analizar los movimientos monitorizados de las extremidades corporales de forma que se pueda hacer un seguimiento durante unas rutinas físicas. Es por ello que la elección de los diferentes tipos de clasificación de los datos se busca que concluya en la obtención de unos buenos resultados.

A la hora de hacer la clasificación de los diferentes ejercicios, se emplearon técnicas de Machine Learning supervisadas. En los algoritmos de Machine Learning se hace uso de datos de entrenamiento (Train) y datos de prueba (Test). En el caso del proyecto, se hizo uso del 75 % de los datos para entrenar el modelo y los restantes para probarlo.

Además, para que los resultados sean coherentes se realiza lo que se denomina validación cruzada (Cross Validation, CV) aleatoria. Dicho término hace referencia a una técnica utilizada para evaluar los resultados de un análisis estadístico y garantizar que sean independientes de la partición entre datos de entrenamiento y prueba. Debido a la gran cantidad de datos que se manejan en el proyecto, se realiza una CV aleatoria con 10 iteraciones.

3.5.1 Tipos de clasificación

Para el correcto estudio de los datos se han determinado tres tipos de clasificación que van a ayudar a evaluar e identificar los ejercicios realizados por los voluntarios.

- **A. Evaluación de los ejercicios:** Consiste en la evaluación de la calidad durante la ejecución del ejercicio, esto es, la clasificación entre una ejecución correcta o incorrecta del mismo.
- **B. Identificación de los ejercicios:** Para identificar un ejercicio entre un catálogo se han analizado dos opciones:
  - **B1. Identificación de los ejercicios bien hechos:** En este tipo de clasificación, se pretende identificar los ejercicios bien realizados incluidos dentro de la tabla de ejercicios propuestos. En dicha identificación, se emplean siete etiquetas, como se puede observar en el bloque B1 de la Figura 3.11, que se corresponden con cada uno de los tipos de ejercicios evaluados.
  - **B2. Identificación de los ejercicios bien y mal hechos:** En esta clasificación, se pretende identificar todos los ejercicios realizados a partir de repeticiones que estén bien y mal ejecutadas, de forma que la variabilidad de movimientos para su clasificación es mucho mayor que en el caso anterior. En esta clasificación hay siete etiquetas que son las correspondientes a cada uno de los ejercicios evaluados, tal y como se representa en el bloque B2 de la Figura 3.11.
- **C. Evaluación e identificación de los ejercicios:** Para evaluar e identificar los ejercicios se hace uso de dos tipos de clasificación:

– **C1. Identificación y evaluación de los ejercicios bien hechos y una clase denominada "wrng":** Con este planteamiento, se pretende identificar los ejercicios bien hechos e identificar si cualquiera de ellos se está ejecutando de manera incorrecta. De esta forma se combinan los planteamientos de identificación de ejercicios, detectando de qué clase es cada movimiento correcto que se está llevando a cabo, con la evaluación de los mismos, determinando si se está realizando alguno de ellos de manera incorrecta. Dentro de esta clasificación hay ocho etiquetas, como se puede observar en el bloque C1 de la Figura 3.11, que son las correspondientes a cada uno de los ejercicios y una nueva etiqueta que denota que es un ejercicio mal realizado (wrng).

– **C2. Identificación y evaluación de los ejercicios bien y mal hechos:** En este tipo de clasificación, el objetivo es identificar todos los ejercicios realizados y, además, determinar si se realizan de forma correcta o incorrecta. Dentro de esta clasificación hay catorce etiquetas, como se puede observar en el bloque C2 de la Figura 3.11, siete de ellas son las correspondientes a los ejercicios bien hechos y las otras siete a los ejercicios mal hechos.

La Figura 3.11 representa un diagrama de bloques en el que se muestran los tres tipos de procedimientos a la hora de clasificar. Dichos procedimientos son independientes entre sí, es por ello que se obtienen resultados diferentes con distinto grado de complejidad y de exactitud.

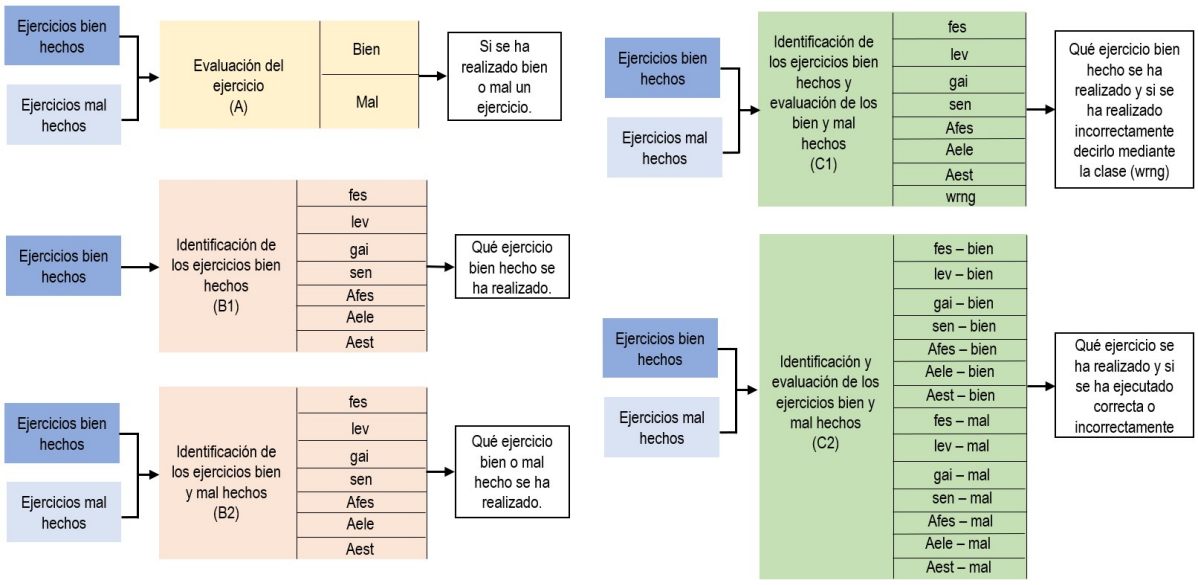


Figura 3.11: Diagrama de bloques en el que se representan las entradas y salidas entre las posibles clasificaciones de los ejercicios.

Para todas las clasificaciones, las características obtenidas de la segmentación por ventana se guardan en una matriz denominada como *wcut.mtz*. En dicha matriz se guardan tanto las características de cada ejercicio como las etiquetas del mismo, es decir, si está bien realizado o mal realizado.

Es por ello, que se crea una matriz en la que se introducen los datos a utilizar para cada tipo de clasificación. A continuación se normalizan los valores de las características contenidas en dicha matriz y se aleatorizan para que los datos no se encuentren ordenados ni por voluntario ni por calidad en la ejecución del ejercicio. Así, se asegura que tanto en los datos de Test como en los de Train se incluyen

muestras de varios voluntarios, varios tipos de identificación y varios tipos de calidad de ejecución.

La calidad de la clasificación se evalúa con el error de clasificación, que representa el número de clases mal predichas con respecto al total de clases sobre los datos de Test, definido en la ecuación (2.19). Como se ha explicado en el apartado 2.2 para realizar un estudio más en profundidad de los resultados a la hora de clasificar, se hace uso, además del error de clasificación, de las métricas de la exactitud (2.20), la especificidad (2.21) y la sensibilidad (2.22) sobre los datos de Test. Dichas métricas son promediadas para el total de las iteraciones de CV.

En los anexos D, E, F, G, H se muestran los códigos creados para evaluar (A), identificar (B1 y B2) y evaluar e identificar los ejercicios (C1 y C2), respectivamente.

3.5.2 Clasificadores aplicados

Para hacer una clasificación correcta de los ejercicios grabados por diferentes voluntarios, la clasificación se ha realizado con algoritmos de Machine Learning Supervisados. Estos algoritmos emplean diferentes características de entrenamiento (*XTrain*) con sus correspondientes etiquetas (*YTrain*). Se han analizado ocho de los algoritmos más utilizados en la literatura, como se explica en el apartado 2.2, y su implementación se ha realizado a través de funciones de Matlab que se detallan a continuación.

- **Algoritmo de Naive Bayes:** Para entrenar el modelo, la función usada tanto para la clasificación binaria como multiclase es la siguiente:

  mdl_NB=fitcnb(XTrain, YTrain)

- **Algoritmo de Naive Bayes con kernel:** Para entrenar el modelo, la función usada en el entorno de Matlab tanto para la clasificación binaria como multiclase es la siguiente:

  mdl_NB_kernel=fitcnb(XTrain, YTrain, "DistributionNames", "kernel");

  Donde *"DistributionNames"* y *"kernel"* son los parámetros que se utilizan para definir que se quiere usar el algoritmo de Naive Bayes con kernel.

- **Algoritmo de SVM (Support Vector Machine)**: Para entrenar el modelo, la función usada en el entorno de Matlab para la clasificación binaria es la siguiente:

  mdl_SVM=fitcsvm(XTrain, YTrain, "Standarize", true);

  Siendo *"Standarize"* y *"true"* son los parámetros que se utilizan para definir que se quiere estandarizar los predictores.

  Por otro lado, en la clasificación multiclase se usa la siguiente función:

  template=templateSVM("Standarize", true);

  mdl_SVM=fitcecoc(XTrain, YTrain, "Learners", template, "FitPosterior", true);

Siendo *"Learners"* y *template* los que indican que se quiere utilizar el algoritmo de SVM sin ningún tipo de kernel. Además, *"FitPosterior"* y *"true"* son los parámetros que transforman los resultados de clasificación en probabilidades de clase, que son devueltas por predict.

El uso de Kernels hace que sea más fácil lidiar con problemas no lineales. Es por ello que se usan tres tipos de Kernel: Kernel lineal, gaussiano y polinómico, que se definen a continuación. Donde *"KernelFunction"* y *"lineal/gaussian/polynomial"* se refiere a los tipos de Kernel de SVM elegidos y, *"KernelScale"* y *auto* son los parámetros que se utilizan para definir que la escala del Kernel se actualice automáticamente.

- **Algoritmo de SVM con kernel lineal:** Para entrenar el modelo, la función usada en el entorno de Matlab para la clasificación binaria es la siguiente:

  mdl_SVM_lin=fitcsvm(XTrain, YTrain, "KernelFunction", "linear", "KernelScale", auto, "Standarize", true);

  En la clasificación multiclase, se emplea el siguiente par de funciones:

  template_lin=templateSVM("Standarize", true, "KernelFunction", "linear", "KernelScale", auto);

  mdl_SVM_lin=fitcecoc(XTrain, YTrain, "Learners", template_lin, "FitPosterior", true);

- **Algoritmo de SVM con kernel gaussiano:** Para entrenar el modelo, la función usada en el entorno de Matlab para la clasificación binaria es la siguiente:

  mdl_SVM_gauss=fitcsvm(XTrain, YTrain, "KernelFunction", "gaussian", "KernelScale", auto, "Standarize", true);

  Mientras que la función usada en la clasificación multiclase es la siguiente:

  template_gauss=templateSVM("Standarize", true, "KernelFunction", "gaussian", "KernelScale", auto);

  mdl_SVM_lin=fitcecoc(XTrain, YTrain, "Learners", template_gauss, "FitPosterior", true);

- **Algoritmo de SVM con kernel polinomial:** Para entrenar el modelo, la función usada en el entorno de Matlab para la clasificación binaria es la siguiente:

  mdl_SVM_pol=fitcsvm(XTrain, YTrain, "KernelFunction", "polynomial", "KernelScale", auto, "Standarize", true);

  La función usada en el entorno de Matlab para el entrenamiento de la clasificación multiclase es la siguiente:

  template_pol=templateSVM("Standarize", true, "KernelFunction", "polynomial", "KernelScale", auto);

  mdlSVM_pol=fitcecoc(XTrain, YTrain, "Learners", template_pol, "FitPosterior", true);

- **Algoritmo de Random Forests:** Para entrenar el modelo, la función usada en el entorno de Matlab para la clasificación binaria es la siguiente:

  mdl_RF=fitcensemble(XTrain, YTrain, "method", "Bag");

  Siendo *"method"* y *"Bag"* el método de configuración de la función *fitcensemble* para que clasifique mediante el algoritmo de Random Forests.

  La función usada en el entorno de Matlab para el entrenamiento de la clasificación multiclase es la siguiente:

  template_RF=templateTree(Reproducible", true);

  mdl_RF=fitcensemble(XTrain, YTrain, "method", "Bag", "NumLearningCycles", 200, "Learners", t);

  Además, *"NumLearningCycles"* determina el número de ciclos de aprendizaje, es decir, el número de árboles de decisión usados en el Random Forest.

- **Algoritmo de KNN (K-Nearest-Neighbours):** Para entrenar el modelo, la función usada en el entorno de Matlab para la clasificación binaria es la siguiente:

  mdl_KNN=fitcknn(XTrain, YTrain, "Standarize", true);

  La función usada en el entorno de Matlab para el entrenamiento de la clasificación multiclase es la siguiente:

  template_KNN=templateKNN("Standarize", true);

  mdl_KNN=fitcecoc(XTrain, YTrain, "Learners", template_KNN, "FitPosterior", true);

  Siendo *"Learners"* y *template_KNN* los que indican que se quiere utilizar el algoritmo de KNN sin ningún tipo de kernel. Además, *"FitPosterior"* y *"true"* son los parámetros que transforman los resultados de clasificación en probabilidades de clase, que son devueltas por predict.

En los anexos B y C, se muestran los scripts de Matlab creados en los que se hace uso de las funciones definidas anteriormente para las clasificaciones estudiadas.

3.5.3 Evaluación de los datos de entrada

La clasificación de los ejercicios se realiza a partir de los datos en crudo de la IMU y también incluyendo como dato adicional los ángulos de Euler correspondientes.

Primero se realiza una clasificación con la información en crudo de la aceleración lineal y la velocidad angular obtenida a partir de las IMUs. Dichas señales se segmentan y, a partir de ellas, se obtienen las características que se introducen en los clasificadores.

De forma paralela, se hace otra clasificación en la que, además de considerar la aceleración lineal y velocidad angular, se introduce la información relativa a los ángulos Roll Y Pitch de las IMUs obtenidos

a partir del UKF propuesto en [10], previamente detallado en el apartado 2.1.1. En la fase de predicción de este UKF, se estiman los ángulos de Euler a partir de la integración de los datos del giróscopo, y en la fase de corrección, se corrigen los ángulos Roll y Pitch con las medidas del acelerómetro cuando el valor de la aceleración medida es similar al de la gravedad. Dichas señales se segmentan de forma similar al caso previo y se obtienen las características que se introducen en los clasificadores.

3.5.4 Evaluación de las posiciones óptimas de las IMUs

Como se ha indicado en el apartado 3.2.2, en la realización de este proyecto se ha hecho uso de cuatro IMUs tanto en los ejercicios de pierna como en los de los brazos.

Lo ideal, en un futuro, es emplear un número menor de IMUs para que la grabación de los movimientos del voluntario sea más cómodo, rápido y el posterior análisis de dichos datos sea mucho más simple e implique menos tiempo. De esta forma, se requieren además menos recursos, tanto materiales, como computacionales.

Es por ello, que se realiza un estudio en el que se comparan para el distinto número de IMUs posibles, es decir de 1 a 4 IMUs, las diferentes posibles posiciones en las que se pueden situar estos sensores a la hora de realizar los movimientos.

Capítulo 4

Resultados obtenidos

En este capítulo se muestran los resultados obtenidos al realizar las clasificaciones según las premisas de evaluación e identificación de ejercicios explicadas en el capítulo 3. En primer lugar se hace uso de la clasificación binaria para evaluar la calidad de un ejercicio en concreto. Por otro lado, se hace uso de la clasificación multiclase para la identificación y la evaluación de los ejercicios dentro de un catálogo de ejercicios creados. Además, dentro de dichas clasificaciones se estudia la influencia del uso de la información de los ángulos de Euler. En ambos casos se evalúan los distintos algoritmos de Machine Learning para proponer el más adecuado para esta aplicación. Por otro lado, se analiza el efecto de las posiciones de las IMUs para optimizar el número de sensores utilizados tal y como se detalla en el apartado 3.5.4.

En este proyecto se han realizado siete ejercicios, explicados con detalle en el apartado 3.1, los cuales se pueden observar en la Figura 4.1.

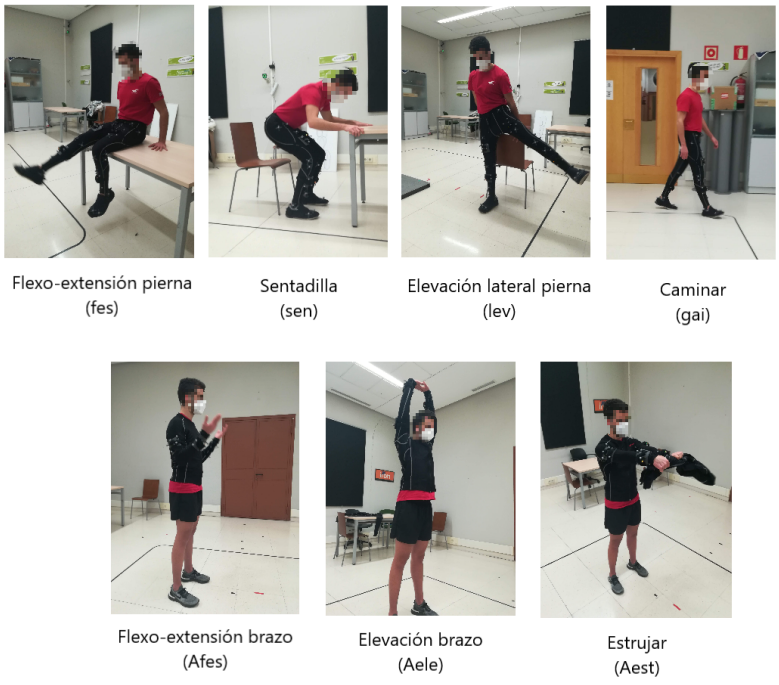


Figura 4.1: Ejercicios elegidos de la rueda de Vivifrail [11] realizados en el laboratorio.

Se han usado cuatro IMUs situadas en las extremidades corporales de los voluntarios como se explica en el apartado 3.2.2. Para realizar el estudio de la optimización de las posiciones de las IMUs, se definen las 15 posibles posiciones de los 4 sensores disponibles en la realización del proyecto. La Tabla 4.1 está compuesta por las 15 posibles posiciones que, además, está divida en tres grupos. El primero de ellos indica las posiciones referidas al uso de una IMU, el segundo grupo indica las posiciones referidas a la utilización de dos IMUs y el último indica las posiciones al utilizar tres y cuatro IMUs, en ese orden.

Tabla 4.1: Equivalencia del número de posición de la IMU con la posición en piernas y brazos. Siendo R, L, T, S, A y F: derecha (right), izquierda (left), muslo (thigh), espinilla (shin), brazo (arm) y antebrazo (forearm), respectivamente.

Nº de posiciones	**Posiciones pierna**	**Posiciones brazo**
1	LT	LA
2	LS	LF
3	RT	RA
4	RS	RF
5	LS-LT	LF-LA
6	LT-RT	LA-RA
7	LT-RS	LA-RF
8	LS-RT	LF-RA
9	LS-RS	LF-RF
10	RS-RT	RF-RA
11	LS-LT-RT	LF-LA-RA
12	LS-RS-RT	LF-RF-RA
13	LT-RS-RT	LA-RF-RA
14	LS-LT-RS	LF-LA-RF
15	LS-LT-RS-RT	LF-LA-RF-RA

Dentro del estudio de cada algoritmo se representa el porcentaje de error de clasificación (2.19) que se obtiene al clasificar los ejercicios con respecto al algoritmo utilizado y la posición de las IMUs. El porcentaje de error de clasificación se compara con tres métricas que ayudan a determinar las mejores posiciones de los sensores en el cuerpo. Estas métricas son la especificidad (2.21), la sensibilidad (2.22) y la exactitud (2.20), denotadas como E, S y Ex, respectivamente.

Con el fin de evaluar las distintas combinaciones de IMUs para encontrar la posición óptima de estas, se estudian dichas métricas según la el número de IMUs utilizadas, es decir, se presentan tres tablas, tanto para los ejercicios de pierna como para los de brazo, que corresponden con la utilización de una, dos y tres o cuatro IMUs, respectivamente.

4.1 Clasificación binaria

Una primera propuesta a la hora de clasificar los datos es la evaluación de la ejecución de un ejercicio. Este tipo de clasificación evalúa la calidad de los ejercicios, de forma que contando con la información previa de qué ejercicio se está realizando, se puede evaluar si se está haciendo acorde al movimiento propuesto o difiere del mismo. En la Figura 4.2 se muestra el diagrama de bloques en el que se introducen los datos del ejercicio a evaluar y se obtiene, como salida, si se ha realizado correcta o incorrectamente.

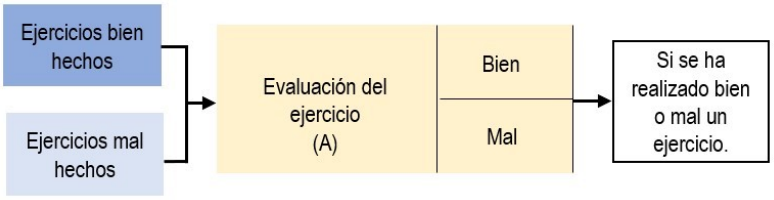


Figura 4.2: Entradas y salidas del bloque de evaluación de los ejercicios.

4.1.1 Evaluación de los ejercicios

Como se ha explicado anteriormente, una de las posibles clasificaciones es la evaluación de cada ejercicio con información de la aceleración lineal y la velocidad angular que miden las diferentes IMUs situadas en las extremidades del cuerpo al realizar los movimientos.

A. Selección de los mejores algoritmos

Se realiza un estudio en el que se hace uso de ocho los algoritmos previamente citados, cuyos resultados se pueden observar en la Tabla 4.2. En dicha Tabla se refleja la media de los errores de clasificación obtenidos al hacer un remuestreo con validación cruzada de 10 iteraciones con respecto a los algoritmos de Machine Learning en los siete diferentes ejercicios realizados. En la primera columna de la Tabla 4.2 se pueden observar los ocho algoritmos utilizados y en la segunda fila, divididos con una doble línea, se encuentran los ejercicios de pierna y los ejercicios de brazo. Además, en la última fila de la Tabla 4.2 se observa el número de conjuntos de características y su etiqueta, denotado como número de datos.

Como se representa en la Tabla 4.2, los resultados obtenidos para las clasificaciones realizadas son diferentes en función del ejercicio evaluado y la clasificación utilizada. Por ejemplo, en el ejercicio de elevación de brazos (Aele) los resultados rondan el 1 % de error, por lo que se puede afirmar que los resultados para dicho ejercicio son muy precisos. Esto es así porque la diferencia entre la elevación de los brazos bien hecha y mal hecha se varía mucho en su realización y, por ello, las características que se obtienen de dichos datos son muy diferentes. Esto se corrobora al comparar ese error de aproximadamente el 1 % con los obtenidos en otros ejercicios, como sentadillas (sen), cuyos valores rondan el 3 % de error, las diferencias entre las repeticiones bien hechas y mal hechas de dicho ejercicio son más difíciles de realizar lo que da lugar a que sus características no sean tan diferentes y, por lo tanto, haya más error a la hora de clasificar. Además, la cantidad de datos introducida en la clasificación es determinante en el resultado obtenido, ya que cuanta más cantidad de datos se introduzca, los resultados son más precisos.

Tabla 4.2: Errores de clasificación obtenidos al hacer la clasificación binaria con la información de la aceleración lineal y la velocidad angular, introduciendo, además, el número datos referido al número de pares de datos para cada clasificación. Los resultados en azul son los cuatro menores errores para la clasificación de cada ejercicio.

	Error de evaluación del ejercicio (%)						
Algoritmo	**fes**	**sen**	**lev**	**gai**	**Afes**	**Aele**	**Aest**
NB	28,00	19,61	26,01	11,78	9,52	3,29	10,92
NB$_k$	18,96	12,22	14,49	12,92	7,28	2,49	9,75
SVM	8,06	8,38	7,16	2,65	2,02	0,61	4,92
SVM$_l$	9,81	9,43	8,18	3,02	4,89	0,75	6,15
SVM$_g$	2,18	2,36	4,14	0,76	3,10	0,34	3,13
SVM$_p$	2,18	2,28	3,25	1,23	2,05	0,35	1,79
RF	2,72	3,85	3,86	3,04	4,70	1,21	3,33
KNN	1,19	1,97	2,40	0,86	2,28	0,47	1,05
Número de datos	1818	930	1872	1025	560	1176	880

A la vista de los resultados se puede considerar que los ejercicios de ejercicio de elevación de pierna (lev) y estrujar (Aest) son los más restrictivos al presentar los errores de clasificación más altos para todos los tipos de clasificación. Por lo tanto, son los ejercicios que se consideran representativos del conjunto para hacer un análisis más exhaustivo que permita identificar al método más adecuado para cubrir el objetivo de este trabajo.

Como se puede observar en la Tabla 4.2 el algoritmo de KNN da los menores errores tanto para lev ($2,4\,\%$) como Aest ($1,05\,\%$), por lo que se puede afirmar que se trata del algoritmo que da mejores resultados. Por otro lado, el segundo algoritmo que da mejores resultados se trata del SVM$_p$ ya que para el ejercicio de lev el error es de $3,25\,\%$ y para el ejercicio de Aest es de $1,79\,\%$. Como se representa en la Tabla 4.2 la diferencia entre el tercer y cuarto mejor algoritmo no se aprecia tan fácilmente, es por ello que se observa para los demás ejercicios de pierna y de brazo y se llega a la conclusión que el tercer mejor algoritmo es el SVM$_g$ ya que sus errores rondan el $2-4\,\%$ en los ejercicios de pierna y un $3\,\%$ en los ejercicios de brazo. Por último, se concluye que el cuarto mejor algoritmo se trata del de RF ya que sus errores rondan el $4\,\%$ en la clasificación de ejercicios de pierna y de brazo.

B. Influencia de la posición de las IMUs

Tras tener determinados cuáles son los cuatro algoritmos que dan mejores resultados para la clasificación binaria con información de la aceleración lineal y la velocidad angular, se comparan dichos algoritmos para la obtención de las posiciones óptimas de las IMUs. Como se ha explicado en el apartado 3.5.4 el objetivo de esta elección de posiciones óptimas es hacer uso de un número menor de IMUs en futuras evaluaciones, lo que da lugar a un requerimiento menor de recursos tanto materiales como computacionales.

Para la elección de las posiciones óptimas de las IMUs, se estudian los errores de clasificación obtenidos para los cuatro mejores algoritmos: SVM$_g$, SVM$_p$, RF y KNN definidos en las Tablas 4.3, 4.4, 4.5 y 4.6, respectivamente. En la primera columna se representan las posiciones de las IMUs definidas en la Ta-

bla 4.1, lo que permite subdividir la tabla en tres cuerpos, uno primero para los resultados utilizando una única IMU, el segundo al utilizar dos IMUs y el tercero al hacer uso de tres y cuatro IMUs. En la segunda fila se representan los ejercicios realizados tanto de pierna como de brazos subdiviendo así las tablas en dos, la parte izquierda representa los ejercicios de miembros inferiores y la derecha de los superiores.

Tabla 4.3: Errores de clasificación binaria del algoritmo SVM gaussiano con respecto a la posición de las IMUs. Clasificación con información de la aceleración lineal y la velocidad angular.

Error de evaluación del ejercicio con SVM_g (%)							
Posición	**fes**	**sen**	**lev**	**gai**	**Afes**	**Aele**	**Aest**
1	3,48	2,52	5,65	1,61	7,09	0,57	4,50
2	4,51	4,16	10,20	1,55	3,25	0,41	3,27
3	5,76	4,32	7,04	1,52	5,95	0,49	3,45
4	4,72	4,28	9,69	0,79	3,41	0,74	3,15
5	2,06	2,48	6,03	0,49	1,58	0,23	3,40
6	1,81	1,96	2,04	1,20	5,22	0,16	3,16
7	2,11	1,83	1,43	0,56	3,00	0,27	3,35
8	0,52	2,06	2,68	0,69	2,03	0,20	2,58
9	1,23	2,13	2,19	0,46	2,68	0,30	1,75
10	2,82	1,63	6,13	0,58	2,78	0,47	3,32
11	0,40	1,83	2,31	0,50	1,79	0,23	2,80
12	0,61	1,30	1,82	0,27	1,77	0,23	2,35
13	1,80	1,87	1,47	0,42	2,64	0,27	3,57
14	0,48	1,58	1,73	0,38	1,79	0,23	3,05
15	0,46	1,46	1,64	0,38	1,45	0,23	3,25

Tabla 4.4: Errores de clasificación binaria del algoritmo SVM polinómico con respecto a la posición de las IMUs. Clasificación con información de la aceleración lineal y la velocidad angular.

Error de evaluación del ejercicio con SVM_p (%)							
Posición	**fes**	**sen**	**lev**	**gai**	**Afes**	**Aele**	**Aest**
1	5,18	2,87	5,57	2,27	6,20	0,54	3,83
2	4,51	4,45	8,03	1,72	2,52	0,50	2,55
3	6,55	4,24	6,20	2,02	5,80	0,69	2,35
4	4,67	5,39	7,77	0,93	1,73	0,83	2,81
5	1,65	1,62	3,52	0,91	0,95	0,27	1,90
6	1,76	1,58	1,60	1,81	4,32	0,29	2,36
7	1,00	1,94	1,08	1,41	1,49	0,30	1,26
8	0,84	1,72	2,07	1,42	1,04	0,23	1,58
9	1,24	2,33	2,17	0,81	0,95	0,30	1,18
10	2,01	1,43	5,46	0,80	1,19	0,43	1,58
11	0,78	1,22	0,99	1,05	0,95	0,13	1,51
12	0,46	1,40	1,45	0,88	0,59	0,13	0,80
13	1,04	1,21	1,12	0,77	1,42	0,13	1,05
14	0,59	1,63	0,97	0,91	0,72	0,30	1,02
15	0,42	1,23	0,80	0,79	0,95	0,17	1,02

Tabla 4.5: Errores de clasificación binaria del algoritmo Random Forests con respecto a la posición de las IMUs. Clasificación con información de la aceleración lineal y la velocidad angular.

Error de evaluación del ejercicio con RF (%)							
Posición	**fes**	**sen**	**lev**	**gai**	**Afes**	**Aele**	**Aest**
1	4,71	3,71	5,21	4,33	4,82	2,11	3,64
2	3,38	5,95	5,81	2,88	7,97	1,22	4,45
3	3,99	6,57	5,34	4,68	6,09	2,17	4,39
4	4,35	5,04	5,46	3,59	5,64	1,67	4,68
5	2,38	2,00	4,13	2,85	4,72	1,32	3,27
6	2,43	4,83	3,46	3,85	6,46	1,47	2,52
7	2,34	3,36	3,38	2,48	4,28	0,65	3,06
8	2,28	2,25	3,25	2,79	4,74	1,27	3,33
9	2,36	2,93	2,76	2,42	4,60	0,95	3,64
10	2,49	3,50	4,79	2,99	4,37	1,43	3,10
11	2,22	4,54	3,17	3,10	2,81	1,06	2,95
12	1,92	3,83	3,16	2,59	3,17	0,75	2,31
13	1,62	3,27	2,65	2,21	3,00	0,35	2,44
14	1,99	2,68	3,11	2,81	3,59	1,06	3,03
15	2,33	3,35	2,18	2,02	4,20	0,67	3,13

Tabla 4.6: Errores de clasificación binaria del algoritmo K-Nearest Neighbours con respecto a la posición de las IMUs. Clasificación con información de la aceleración lineal y la velocidad angular.

Error de evaluación del ejercicio con KNN (%)							
Posición	**fes**	**sen**	**lev**	**gai**	**Afes**	**Aele**	**Aest**
1	1,57	2,92	4,30	1,63	2,87	1,24	1,34
2	1,68	3,12	4,08	1,69	3,99	0,61	2,39
3	1,83	3,63	4,54	1,66	2,85	1,05	1,46
4	2,75	2,11	4,51	1,35	2,46	0,84	2,42
5	0,72	1,79	2,81	0,61	2,35	0,13	1,24
6	1,22	1,96	1,47	0,77	2,24	0,79	0,76
7	1,24	1,86	1,51	0,54	1,40	0,10	0,82
8	0,91	1,78	1,84	0,82	3,00	0,44	0,59
9	2,09	1,74	2,02	0,47	2,75	0,20	0,87
10	0,56	1,52	2,78	0,86	1,00	0,50	0,95
11	0,72	1,22	1,21	0,42	2,40	0,23	0,58
12	0,46	1,18	1,47	0,48	1,89	0,37	0,36
13	0,68	1,86	1,30	0,77	0,77	0,17	0,82
14	0,96	1,35	1,09	0,33	2,29	0,20	0,67
15	0,46	1,47	1,10	0,48	1,86	0,23	0,46

Como se puede observar en las Tablas 4.3, 4.4, 4.5 y 4.6 referidas a los errores de clasificación de los algoritmos de SVM_g, SVM_p, RF y KNN, respectivamente, los errores varían según la posición de cada IMU y el ejercicio realizado. Como se ha explicado anteriormente, el error entre ejercicios varía según la diferencia existente entre las repeticiones bien hechas y las mal hechas, además, dicho error depende de la posición en la que se sitúen las IMUs. Normalmente, cuanta mayor cantidad de IMUs estén situadas sobre las extremidades, menor error de clasificación y, por consiguiente, se obtienen mejores resultados.

En las Tablas 4.3, 4.4, 4.5 y 4.6 se puede observar que el algoritmo que mejor funciona cuando se hace uso de un número menor de IMUs es el algoritmo de KNN, ya que ofrece errores entre un $3 - 0,5\,\%$, mientras que el algoritmo de RF ofrece errores en torno al $5\,\%$. En cambio, para el uso de más de una IMU, los resultados son bastante parecidos entre el algoritmo de SVM_g y KNN y es difícil diferenciar cuál es mejor al comparar entre el algoritmo SVM_p y RF.

Para comparar los diferentes algoritmos y concluir cuál de ellos ofrece mejores características se hace uso de las curvas ROC. Como se ha explicado en el apartado 2.2, la curva ROC es la representación de la *proporción de verdaderos positivos o sensiblidad* frente a la *proporción de falsos positivos o* 1-*especificidad*, por lo que observando cómo dicha gráfica evoluciona se puede explicar cómo de bueno es un algoritmo u otro.

En la Figura 4.3, se representan las curvas ROC de los algoritmos de NB, SVM_g, SVM_p, RF y KNN al hacer uso de las cuatro IMUs utilizadas en las grabaciones. Los resultados de NB se usan como base para destacar el mejor funcionamiento del resto de algoritmos. Como se puede observar en la Figura 4.3, las curvas de los algoritmos SVM_g, SVM_p, RF y KNN son bastante cercanas a la óptima ya que sus valores

de sensibilidad están por encima del 95 % y sus valores de 1-especificidad están por debajo del 5 %. Se puede concluir que, dentro de dichos cuatro algoritmos el que ofrece peores resultados es el RF, y le sigue SVM_p. Por el contrario, los algoritmos SVM_g y KNN tienen unas curvas ROC muy similares. Puesto uno de los algoritmos que ofrece mejores resultados es KNN, de aquí en adelante, se hace uso, simplemente, de las métricas de dicho algoritmo para discutir cuáles son las mejores posiciones de las IMUs sobre las extremidades del cuerpo.

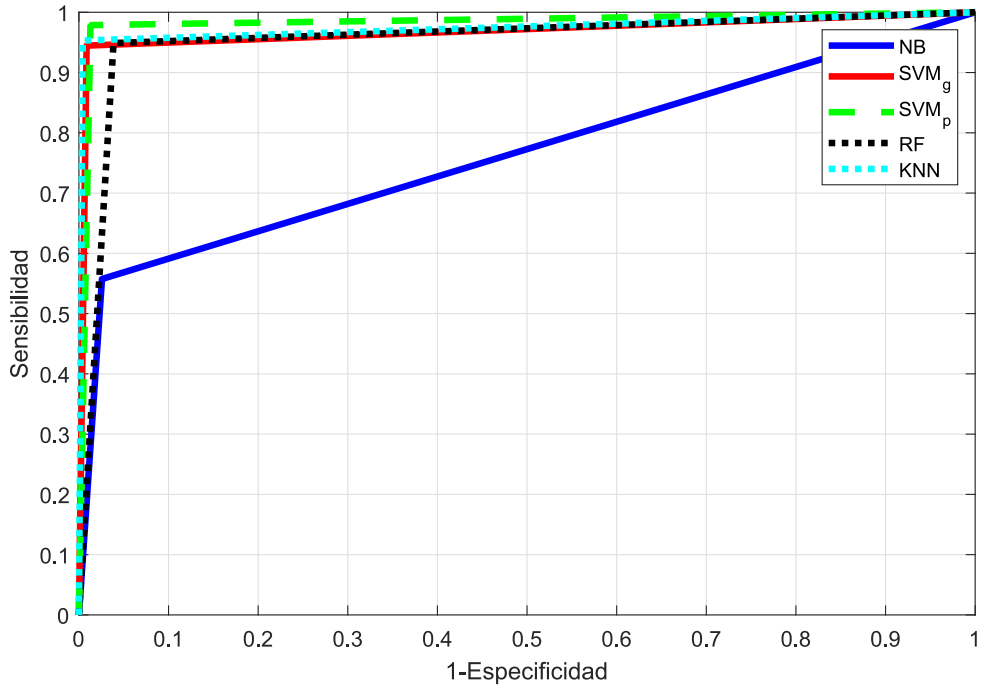


Figura 4.3: Curvas ROC obtenidas de los algoritmos SVM_g, SVM_p, RF y KNN, con respecto a los resultados de NB, al hacer la clasificación con información de la velocidad angular y aceleración lineal.

En las posiciones en las que se hace uso de dos, tres y cuatro sensores, los resultados varían en un 1 % entre dichas posiciones, lo que da lugar a una leve diferencia entre el uso de más de una IMU e implica una difícil decisión entre cuáles son las posiciones óptimas. Para ayudar a obtener dichas posiciones, se hace uso de las métricas de especificidad, sensibilidad y exactitud expuestas anteriormente y definidas en el apartado 2.2.

En la Tabla 4.7 se representan las métricas obtenidas al hacer uso de una sola IMU en los ejercicios de pierna. Como se ha explicado anteriormente, se ha escogido el ejercicio de elevación lateral como ejercicio representativo de los ejercicios de pierna.

Como se representa en la Tabla 4.7 el valor de la sensibilidad está en torno al $98\,\% - 99\,\%$, lo que indica que la detección de una repetición mal hecha es casi una detección perfecta. Además, se puede observar que la especificidad tiene valores en torno al 93 %, dichos porcentajes son muy buenos aunque son más bajos que los de la sensibilidad. Dichos resultados indican que se detectan más falsos positivos que falsos negativos lo que da lugar a una clasificación, a veces, de casos positivos en los casos erróneos. La diferencia de porcentaje entre la sensibilidad y la especificidad se debe a que un ejercicio que se realiza bien tiene pocas variantes ya que su realización está estipulada, mientras que un ejercicio que se realiza mal tiene muchas más variaciones que dependen de la persona que lo realice, cómo lo realice y en las condiciones en

Tabla 4.7: Especificidad, sensibilidad y exactitud de las posiciones en las que se hace uso sólo de una IMU en los ejercicios de pierna. Clasificación binaria de KNN con información de la velocidad angular y aceleración lineal.

(%) 1 IMU. PIERNA			
Posición	**E**	**S**	**Ex**
1 LT	93,00	98,54	95,75
2 LS	93,34	98,67	95,98
3 RT	92,74	98,32	95,51
4 RS	92,53	98,58	95,53

las que lo realice. Por último, en la Tabla 4.7 se detalla que los valores de la exactitud están por encima del 95 % en todos los casos, lo que indica que los datos de los ejercicios se han clasificado correctamente por encima de dicho porcentaje.

Como se puede observar, las diferencias entre las métricas de las cuatro posiciones representadas en la Tabla 4.7 varían, como máximo, en un $0,5$ % entre ellas. Esto indica que, en el caso de sólo utilizar una IMU en la monitorización de los ejercicios, sería indiferente escoger una posición de entre las cuatro posibles en la que situar una IMU en los miembros inferiores, ya que los resultados de dicha monitorización serían similares.

En la Tabla 4.8 se representan las métricas obtenidas al hacer uso de dos IMUs en los ejercicios de pierna. Como se representa, al añadir una IMU en la realización de los ejercicios, los resultados mejoran ya que los valores de la especificidad aumentan en un 4 %, lo que da lugar a una mejor detección de las repeticiones mal hechas que en el caso de utilizar solo una IMU. Además la exactitud aumenta en un 3 % y la sensibilidad en un 1 %, esto es así porque los valores obtenidos de la sensibilidad en la Tabla 4.7 ya son muy altos, casi del 99 %, y son difíciles de mejorar.

Tabla 4.8: Especificidad, sensibilidad y exactitud de las posiciones en las que se hace uso de dos IMUs en los ejercicios de pierna. Clasificación KNN con información de la velocidad angular y aceleración lineal.

(%) 2 IMUs. PIERNA			
Posición	**E**	**S**	**Ex**
5 LS-LT	95,54	98,92	97,22
6 LT-RT	97,41	99,70	98,55
7 LT-RS	97,20	99,83	98,50
8 LS-RT	96,60	99,78	98,18
9 LS-RS	96,35	99,70	98,01
10 RS-RT	95,42	99,10	97,24

Para explicar los resultados obtenidos al hacer uso de dos IMUs, se divide dicha comparación en dos, teniendo por un lado las posiciones de las IMUs en el caso en el que se sitúen sobre la misma pierna y, por otro lado, aquellas posiciones en las que las IMUs estén cruzadas, es decir, una en el muslo y otra en la espinilla, o paralelas, estando ambas en los muslos o ambas en las espinillas.

La primera comparación que se realiza es aquella en la que se hace uso de dos IMUs situadas en la

misma pierna. Dichas posiciones son la 5 y la 10 en la que se representan las IMUs sobre la espinilla y el muslo en la pierna izquierda y derecha, respectivamente. En dichas posiciones los resultados son bastante similares ya que las métricas están en torno al 95 % en el caso de la especificidad, al 99 % en el caso de la sensibilidad y al 97 % en el caso de la exactitud, en ambos casos. Como se puede observar los valores de las métricas de dichas posiciones son mayores que las obtenidas en la Tabla 4.7, donde sólo se utiliza una sola IMU, ya que en ambos casos se hace uso de IMUs situadas sobre una misma pierna. Tal y como se detalla, los resultados son mejores al añadir una IMU más sobre la pierna ya que las métricas mejoran, normalmente, en un 2 %.

La segunda comparación propuesta es aquella en la que se hace uso de las IMUs cruzadas sobre las piernas, que se tratan de las posiciones 8 y 7, LS-RT y RS-LT, respectivamente, o en paralelo, que son las posiciones 6 y 9, LT-RT y LS-RS, respectivamente. Los resultados obtenidos en las posiciones cruzadas son prácticamente iguales, al igual que ocurre en el caso de las posiciones en paralelo, en las que sus resultados son bastante similares. Entre estas cuatro posibles posiciones no hay una diferencia significativa entre los resultados ya que en el caso de la sensibilidad rondan todos el 100 %, en el caso de la especificidad rondan el 97 % y en el caso de la exactitud el 98 %.

Es destacable que, si se hace uso de dos IMUs situadas en la misma pierna, caso de las posiciones 5 y 10, las métricas estudiadas son un 1 % menores que en las posiciones en las que las IMUs están cruzadas. Esto se debe a que monitorizando una única pierna, si hay errores en el movimiento en la pierna contraria, estos no se detectan, aumentando de esta forma el número de errores en la clasificación. Es por ello que, si se hace uso de dos IMUs, lo ideal sería colocarlas una en una pierna y otra en la otra, independientemente de la posición en la que se sitúen.

En la Tabla 4.9 se representan las métricas obtenidas al hacer uso de tres y cuatro IMUs en los ejercicios de pierna. En este caso los valores de la exactitud, sensibilidad y especificidad mejoran en un $0,5$ %, en el mejor de los casos, con respecto al uso de dos IMUs, representados en la Tabla 4.8.

Tabla 4.9: Especificidad, sensibilidad y exactitud de las posiciones en las que se hace uso de tres y cuatro IMUs en los ejercicios de pierna. Clasificación KNN con información de la velocidad angular y aceleración lineal.

(%) 3 y 4 IMUs. PIERNA			
Posición	**E**	**S**	**Ex**
11 LS-LT-RT	97,92	99,70	98,80
12 LS-RS-RT	97,37	99,74	98,55
13 LT-RS-RT	97,79	99,66	98,72
14 LS-LT-RS	98,01	99,87	98,93
15 LS-LT-RS-RT	97,96	99,87	98,91

Esto indica que el cambio de dos a tres o cuatro IMUs no es muy representativo, lo que da lugar a que se acote el uso de dichos sensores. Puesto que lo ideal en una monitorización es el empleo del menor número de sensores posible, puede considerarse que el número óptimo de sensores para esta aplicación es dos, situados de forma cruzada o paralela.

Tras haber discutido cuáles son las mejores posiciones de las IMUs en los ejercicios de pierna, se procede

a definir dichas posiciones óptimas pero en los ejercicios de brazo. Como se ha explicado anteriormente, el ejercicio de referencia en los ejercicios de brazo es el de estrujar. En la Tabla 4.10 se representan las métricas obtenidas al hacer uso de una sola IMU en los ejercicios de brazo.

Tabla 4.10: Especificidad, sensibilidad y exactitud de las posiciones en las que se hace uso sólo de una IMU en los ejercicios de brazos. Clasificación KNN con información de la velocidad angular y aceleración lineal.

(%) 1 IMU. BRAZO			
Posición	**E**	**S**	**Ex**
1 LA	97,61	99,72	98,64
2 LF	95,75	99,53	97,59
3 RA	97,17	100,00	98,55
4 RF	95,31	99,91	97,55

Al igual que ocurre en los ejercicios de pierna, los valores de la sensibilidad están en torno al $99-100\,\%$ y los valores de especificidad son más bajos, cerca del $96\,\%$. Esto da lugar a que la exactitud de la clasificación esté en torno al $98\,\%$. Como se puede observar, las diferencias entre las métricas de las cuatro posiciones representadas en la Tabla 4.10 difieren en un $1\,\%$, lo que indica que es indiferente la posición en la que situar la IMU si se hace uso solo de una de ellas.

En el caso de los ejercicios de brazos los valores de las métricas mejoran en un $4\,\%$ con respecto a los de las piernas, esto es así porque las diferencias entre las repeticiones bien hechas y mal hechas son más fáciles de realizar que en el caso de las piernas. Los movimientos de los ejercicios de brazos son simétricos cuando se hacen bien, siendo más restringidos, mientras que en los de piernas esto no ocurre. Estas variaciones dan lugar a diferencias más notables entre las características de los brazos que en las de de las piernas.

En la Tabla 4.11 se representan las métricas obtenidas al hacer uso de dos IMUs. Como se puede observar, al igual que ocurre en los ejercicios de pierna, si se hace uso de una IMU más que en la Tabla 4.10, los resultados obtenidos mejoran.

Tabla 4.11: Especificidad, sensibilidad y exactitud de las posiciones en las que se hace uso de dos IMUs en los ejercicios de brazos. Clasificación KNN con información de la velocidad angular y aceleración lineal.

(%) 2 IMU. BRAZO			
Posición	**E**	**S**	**Ex**
5 LF-LA	97,79	99,72	98,73
6 LA-RA	98,49	100,00	99,23
7 LA-RF	98,41	100,00	99,18
8 LF-RA	98,85	100,00	99,41
9 LF-RF	98,32	100,00	99,14
10 RF-RA	98,14	100,00	99,05

Como ocurre en los ejercicios de pierna, las posiciones de las IMUs en los ejercicios de brazo se pueden colocar sobre el mismo brazo o sobre ambos brazos, cruzadas o en paralelo. En el caso de hacer uso de dos IMUs, como se puede observar en la Tabla 4.11, los resultados son similares ya que la sensibilidad

es del 100 %, la especificidad está en torno al 98 % y la exactitud ronda el 99 %. Es por ello que, si se hace uso de dos IMUs situadas en el mismo brazo, el error de clasificación aumenta y las métricas son ligeramente peores. Para detectar si está moviendo dicho brazo el error es inapreciable, pero si el brazo que se está moviendo es el otro en la que no hay situada ninguna IMU, dichos sensores no lo detectan y se produce un error más significativo. Es por ello que, si se hace uso de dos IMUs, aunque las métricas solo mejoren ligeramente, lo ideal es colocarlas cruzadas o en paralelo en ambos brazos.

En la Tabla 4.12 se representan las métricas obtenidas al hacer uso de tres y cuatro IMUs en los ejercicios de brazo.

Tabla 4.12: Especificidad, sensibilidad y exactitud de las posiciones en las que se hace uso de tres y cuatro IMUs en los ejercicios de brazos. Clasificación KNN con información de la velocidad angular y aceleración lineal.

(%) 3 y 4 IMUs. BRAZO			
Posición	**E**	**S**	**Ex**
11 LF-LA-RA	98,85	100,00	99,41
12 LF-RF-RA	99,29	100,00	99,64
13 LA-RF-RA	98,41	100,00	99,18
14 LF-LA-RF	98,67	100,00	99,32
15 LF-LA-RF-RA	99,11	100,00	99,55

Comparando esta Tabla con respecto a la Tabla 4.11, los resultados son bastante similares ya que la sensibilidad, especificidad y exactitud rondan, en ambos casos, el 100 %, 99 % y 99 %, respectivamente. Esto indica que el cambio de uso de dos a tres o cuatro IMUs no es muy representativo. Esto implica que, igual que en el caso de las piernas, el número óptimo de sensores para esta aplicación es dos, quedando situadas, como se ha explicado anteriormente, de forma cruzada o paralela.

4.1.2 Evaluación de los ejercicios con información de los ángulos de Euler

La siguiente clasificación realizada dentro de la clasificación binaria es la que hace uso de la información de la aceleración lineal, la velocidad angular y los ángulos de Euler. El fin de esta clasificación es comparar los resultados con respecto a la clasificación realizada en el apartado 4.1.1 y determinar si la introducción de la información relativa a los ángulos mejora a la hora de clasificar los ejercicios.

A. Selección de los mejores algoritmos

En la Tabla 4.13 se representan las medias de los errores de las clasificaciones binarias de los siete ejercicios establecidos, en la cual se sigue el mismo criterio que en el apartado previo.

En la Tabla 4.13, los resultados coloreados en azul se corresponden con los cuatro menores errores para cada ejercicio. Como ocurre en la sección 4.1.1, los algoritmos que ofrecen mejores resultados son KNN y SVM_p, cuyos errores rondan el 2 %, seguido de SVM_g y RF, con errores de 3 % y 4 %, respectivamente.

En la Figura 4.4 se representan las curvas ROC de los algoritmos de NB, SVM_g, SVM_p, RF y KNN haciendo uso de las cuatro IMUs utilizadas al grabar los ejercicios, para estimar los métodos más adecuados

Tabla 4.13: Errores de clasificación obtenidos al hacer la clasificación binaria con la información de la aceleración lineal, la velocidad angular y los ángulos de Euler, introduciendo el número de datos referido al número de pares de datos para cada clasificación. Los resultados en azul son los cuatro menores errores para la clasificación de cada ejercicio.

	Error de evaluación del ejercicio (%)						
Algoritmo	**fes**	**sen**	**lev**	**gai**	**Afe**	**Aele**	**Aes**
NB	28,68	19,09	25,52	11,70	7,91	3,53	10,61
NB$_k$	19,50	11,58	14,10	12,71	5,79	2,62	9,79
SVM	7,92	8,32	7,19	2,73	1,84	0,86	4,74
SVM$_l$	9,87	9,12	8,29	3,04	3,86	1,06	5,74
SVM$_g$	2,35	2,26	4,09	0,92	2,45	0,28	2,99
SVM$_p$	2,05	2,23	3,40	1,45	1,70	0,49	1,41
RF	2,73	3,91	4,03	3,38	4,53	1,44	2,92
KNN	1,23	2,30	2,66	0,87	1,91	0,51	0,93
Número de datos	1818	930	1872	1025	560	1176	880

a nuestros objetivos. Además, las curvas de estos últimos cuatro algoritmos son bastante cercanas a lo óptimo ya que, como ocurría en el apartado 4.1.1 sus valores de sensibilidad están por encima del 95 % y los de 1-especificidad están por debajo del 5 %. Es por ello que se puede concluir que, de los cuatro mejores algoritmos, el algoritmo que ofrece peores resultados es el RF, pero las diferencias entre las curvas ROC de los algoritmos SVM$_g$, SVM$_p$ y KNN no son significativas. Para comparar las métricas obtenidas entre la clasificación realizada con y sin ángulos de Euler, se utiliza el algoritmo KNN ya que es el que se utiliza a la hora de estudiar las mejores posiciones cuando se realiza la clasificación sin información de los ángulos.

B. Influencia de la posición de las IMUs

Tras tener determinado cuál es el algoritmo que da mejores resultados para la clasificación binaria con información de la aceleración lineal, la velocidad angular y los ángulos de Euler, se definen las posiciones óptimas de las IMUs tanto en los ejercicios de pierna como en los ejercicios de brazo.

En la Tabla 4.14 se representan las métricas obtenidas al hacer uso de una sola IMU en los ejercicios de pierna. Como se ha explicado anteriormente, se ha escogido el ejercicio de elevación lateral de pierna como ejercicio representativo de los ejercicios de pierna.

En la Tabla 4.14 se representa la comparativa entre los resultados obtenidos para la clasificación haciendo uso de los ángulos de Euler y en la que no se hace uso de dichos ángulos. Como se puede observar, los valores de las tres métricas son similares en ambas clasificaciones ya que la sensibilidad, exactitud y especificidad están en torno a 98 %, 95 % y 93 %, respectivamente. Por lo que se puede concluir que añadir la información de los ángulos de Euler en la detección de los ejercicios no mejora la clasificación.

Las diferencias entre las métricas de las cuatro posiciones representadas en la Tabla 4.14 no difieren mucho. Esto indica, como ocurría en el apartado 4.1.1, que en el caso de solo utilizar una IMU en la

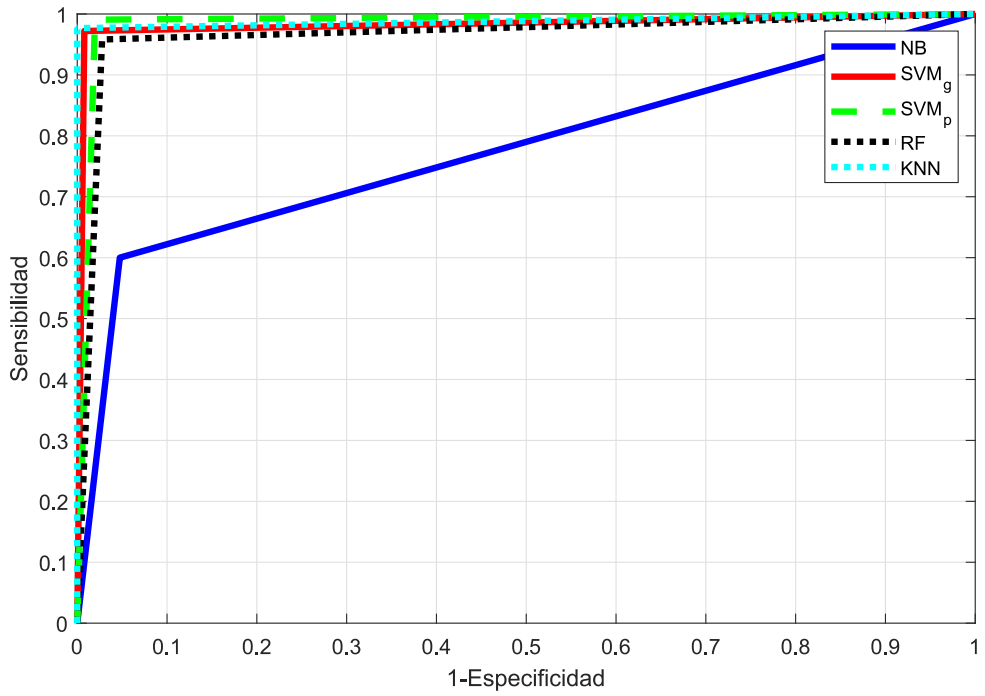


Figura 4.4: Curvas ROC obtenidas de los algoritmos SVM_g, SVM_p, RF y KNN, con respecto a los resultados de NB, al hacer la clasificación con información de la velocidad angular, de la aceleración lineal y los ángulos de Euler.

monitorización de los ejercicios, sería indiferente escoger una posición de entre las cuatro posibles en la que situar una IMU, ya que los resultados de la exactitud de la clasificación rondaría el 95 %.

En la Tabla 4.15 se representan las métricas obtenidas al hacer uso de dos IMUs. Como se puede observar, añadir una IMU en la realización de un ejercicio, dará lugar a la obtención de mejores resultados ya que se incrementan los valores de las métricas, aproximadamente, en un 3 %.

Para explicar los resultados se divide la discusión en dos para comparar las posiciones de las IMUs en el caso en el que se sitúen sobre la misma pierna y, también, para comparar aquellas posiciones en las que las IMUs estén cruzadas o paralelas. Los resultados obtenidos en la Tabla 4.15 corroboran lo discutido en el apartado 4.1.1, ya que para las posiciones en las que se hace uso de dos IMUs situadas en la misma

Tabla 4.14: Especificidad, sensibilidad y exactitud de las posiciones en las que se hace uso sólo de una IMU en los ejercicios de pierna. Clasificación KNN con información de la velocidad angular, la aceleración lineal y los ángulos de Euler.

(%) 1 IMU. PIERNA						
Posición	**Clas. sin ángulos**			**Clas. con ángulos**		
	E	**S**	**Ex**	**E**	**S**	**Ex**
1 LT	93,00	98,54	95,75	92,28	98,15	95,19
2 LS	93,34	98,67	95,98	92,83	98,88	95,83
3 RT	92,74	98,32	95,51	93,17	98,45	95,79
4 RS	92,53	98,58	95,53	92,41	98,79	95,58

Tabla 4.15: Especificidad, sensibilidad y exactitud de las posiciones en las que se hace uso de dos IMUs en los ejercicios de pierna. Clasificación KNN con información de la velocidad angular, la aceleración lineal y los ángulos de Euler.

(%) 2 IMUs. PIERNA						
Posición	**Clas. sin ángulos**			**Clas. con ángulos**		
	E	**S**	**Ex**	**E**	**S**	**Ex**
5 LS-LT	95,54	98,92	97,22	94,87	99,27	97,05
6 LT-RT	97,41	99,70	98,55	96,69	99,57	98,12
7 LT-RS	97,20	99,83	98,50	95,76	99,66	97,69
8 LS-RT	96,60	99,78	98,18	96,18	99,66	97,91
9 LS-RS	96,35	99,70	98,01	95,76	99,74	97,74
10 RS-RT	95,42	99,10	97,24	95,16	98,97	97,05

pierna (LS-LT y RS-RT) los resultados son ligeramente inferiores que en las posiciones en las que se encuentran las IMUs cruzadas o paralelas.

Como se puede observar en la Tabla 4.15, los valores de las métricas entre la clasificación haciendo uso de los ángulos difieren frente a no usarlos, normalmente, en un $0,5\,\%$. Aunque los resultados difieran en un $0,5\,\%$ no se puede considerar que una clasificación sea mejor o peor que otra ya que se trata de un porcentaje muy pequeño de diferencia.

Al igual que ocurría en el apartado 4.1.1, los resultados obtenidos al usar las dos IMUs situadas en la pierna derecha (RT y RS) o en la izquierda (LT y LS) son bastante similares ya que las métricas están en torno al $96\,\%$ en el caso de la especificidad, al $99\,\%$ en el caso de la sensibilidad y al $98\,\%$ en el caso de la exactitud. Además, los resultados obtenidos en las posiciones de las IMUs en las que se encuentran cruzadas (LT-RS y LS-RT) son prácticamente iguales en ambas ya que en el caso de la sensibilidad, de la exactitud y de la sensibilidad rondan el $100\,\%$, $98\,\%$ y $97\,\%$, respectivamente. Por su lado cuando las IMUs se sitúan en paralelo (LT-RT y LS-RS) los resultados son muy parecidos entre sí y, además, rondan los mismos porcentajes que los explicados en el caso de estar cruzadas. Si se hace uso de dos IMUs que están situadas en la misma pierna la detección de algunos ejercicios no es tan eficaz (aproximadamente un $97\,\%$ de exactitud), como empleando las IMUs en distintas piernas, donde la precisión aumenta alrededor del $1\,\%$. Es por ello que, si se hace uso de dos IMUs, lo ideal es colocarlas una en una pierna y otra en la otra, independientemente de la posición en la que se sitúen.

En la Tabla 4.16 se representan las métricas obtenidas al hacer uso de tres y cuatro IMUs en los ejercicios de pierna.

Al igual que ocurre en el caso de la utilización de dos IMUs, los resultados entre la clasificación haciendo uso de la información de los ángulos de Euler no hace que los resultados mejoren. Esto es debido a que los resultados obtenidos sin la información de dichos ángulos ya es muy buena y es muy difícil mejorarlos.

Comparando esta Tabla con la Tabla 4.15, los resultados son bastante similares, ya que la sensibilidad,la exactitud y la exactitud rondan el $100\,\%$, $98\,\%$ y $97\,\%$, respectivamente. Esto indica que el cambio de uso de dos a tres o cuatro IMUs no es muy representativo, tal y como ocurre en la clasificación con los

Tabla 4.16: Especificidad, sensibilidad y exactitud de las posiciones en las que se hace uso de tres y cuatro IMUs en los ejercicios de pierna. Clasificación KNN con información de la velocidad angular, la aceleración lineal y los ángulos de Euler.

(%) 3 y 4 IMUs. PIERNA						
Posición	**Clas. sin ángulos**			**Clas. con ángulos**		
	E	**S**	**Ex**	**E**	**S**	**Ex**
11 LS-LT-RT	97,92	99,70	98,80	97,33	99,83	98,57
12 LS-RS-RT	97,37	99,74	98,55	97,28	99,66	98,46
13 LT-RS-RT	97,79	99,66	98,72	97,24	99,61	98,42
14 LS-LT-RS	98,01	99,87	98,93	97,20	99,96	98,57
15 LS-LT-RS-RT	97,96	99,87	98,91	97,62	99,74	98,68

datos en crudo de las IMUs, presentada en la sección 4.1.1. Es por ello que se ha considerado que el número óptimo de sensores para esta aplicación es dos, situadas, como se ha explicado anteriormente, en las piernas de forma cruzada o paralela.

A la hora de elegir las posiciones óptimas de los brazos se realiza una comparativa entre la clasificación con y sin ángulos de Euler entre el ejercicio de estrujar, ya que se trata del más restrictivo y sus resultados son extrapolables al resto de ejercicios de brazo.

En la Tabla 4.17 se representan las métricas obtenidas al hacer uso de una sola IMU en los ejercicios de brazo.

Tabla 4.17: Especificidad, sensibilidad y exactitud de las posiciones en las que se hace uso sólo de una IMU en los ejercicios de brazos. Clasificación KNN con información de la velocidad angular, la aceleración lineal y los ángulos de Euler.

(%) 1 IMU. BRAZO						
Posición	**Clas. sin ángulos**			**Clas. con ángulos**		
	E	**S**	**Ex**	**E**	**S**	**Ex**
1 LA	97,61	99,72	98,64	97,92	99,81	98,82
2 LF	95,75	99,53	97,59	95,31	99,43	97,27
3 RA	97,17	100,00	98,55	97,48	100,00	98,68
4 RF	95,31	99,91	97,55	96,79	99,90	98,27

Como se puede observar en la Tabla 4.17, al realizar la clasificación con los ángulos mejora en un $0,5\,\%$ con respecto a la clasificación sin ángulos. Al igual que ocurría en el caso de los ejercicios de pierna, dicha mejora no es notable ya que los resultados obtenidos al hacer la clasificación sin ángulos ya era buena y no da lugar a una gran mejora de los resultados.

Los valores de la sensibilidad están en torno al $99 - 100\,\%$ y los valores de especificidad están cerca del $96\,\%$. Como se puede observar, las diferencias entre las métricas de las cuatro posiciones representadas en la Tabla 4.17 difieren en un $1\,\%$ lo que indica que colocar un IMU en el brazo o antebrazo del brazo izquierdo o derecho es indiferente, ya que se obtienen muy buenos resultados cuya precisión es cercana al

99 %.

En la Tabla 4.18 se representan las métricas obtenidas al hacer uso de dos IMUs en los ejercicios de brazo. Así, al añadir una IMU en la realización de los ejercicios de brazo, da lugar a la obtención de mejores resultados, como ocurre, también, con los ejercicios de pierna.

Tabla 4.18: Especificidad, sensibilidad y exactitud de las posiciones en las que se hace uso de dos IMUs en los ejercicios de brazos. Clasificación KNN con información de la velocidad angular, la aceleración lineal y los ángulos de Euler.

(%) 2 IMUs. BRAZO						
Posición	**Clas. sin ángulos**			**Clas. con ángulos**		
	E	**S**	**Ex**	**E**	**S**	**Ex**
5 LF-LA	97,79	99,72	98,73	97,22	100,00	98,55
6 LA-RA	98,49	100,00	99,23	98,87	100,00	99,41
7 LA-RF	98,41	100,00	99,18	98,70	100,00	99,32
8 LF-RA	98,85	100,00	99,41	98,78	100,00	99,36
9 LF-RF	98,32	100,00	99,14	98,09	100,00	99,00
10 RF-RA	98,14	100,00	99,05	98,78	100,00	99,36

Las posiciones de las IMUs en los ejercicios de brazo se pueden colocar sobre el mismo brazo, cruzadas o en paralelo. En la Tabla 4.18, se representa la comparativa al entre la clasificación con y sin ángulos de Euler. Como ha ocurrido hasta ahora, los valores son bastante similares ya que tanto para la sensibilidad, especificidad y exactitud, dichos valores varían en un $0,2$ %, en el mayor de los casos, entre ambas clasificaciones.

En el caso de hacer uso de dos IMUs, como se puede observar en la Tabla 4.18, los resultados son prácticamente los mismos ya que la sensibilidad es del 100 %, la especificidad está en torno al 98 % y la exactitud ronda el 99 %. Aunque los resultados obtenidos de las posiciones de las IMUs que se encuentran sobre un mismo brazo (5 y 10) están por encima del 98 %, lo ideal es colocarlas cruzadas o en paralelo en los dos brazos. Esto es así porque para detectar correctamente los movimientos de ambos brazos es necesario situar una IMU en cada brazo ya que si se mueve solo un brazo y no tiene IMUs, dicho movimiento no estaría monitorizado.

En la Tabla 4.19 se representan las métricas obtenidas al hacer uso de tres y cuatro IMUs en los ejercicios de brazo. En este caso los resultados de la exactitud, sensibilidad y especificidad prácticamente no varían con respecto al uso de dos IMUs.

Como se detalla en la Tabla 4.19 la diferencia entre ambos tipos de clasificación considerando o no los ángulos de Euler es inexistente, ya que se obtienen valores de especificidad, sensibilidad y exactitud del 99 %, 100 % y 99 % en ambos casos. Además, al comparar esta Tabla con respecto a la Tabla 4.18, los resultados son bastante similares, lo que indica que el cambio de uso de dos a tres o cuatro IMUs no es muy representativo, lo que da lugar a que se acote el uso de dichos sensores. Es por ello que se considera que el número óptimo de sensores para esta aplicación usando los ángulos de Euler es dos, situadas en los brazos de forma cruzada o paralela.

Tabla 4.19: Especificidad, sensibilidad y exactitud de las posiciones en las que se hace uso de tres y cuatro IMUs en los ejercicios de brazos. Clasificación KNN con información de la velocidad angular, la aceleración lineal y los ángulos de Euler.

(%) 3 y 4 IMUs. BRAZO						
Posición	**Clas. sin ángulos**			**Clas. con ángulos**		
	E	**S**	**Ex**	**E**	**S**	**Ex**
11 LF-LA-RA	98,85	100,00	99,41	99,05	100,00	99,50
12 LF-RF-RA	99,29	100,00	99,64	98,96	100,00	99,45
13 LA-RF-RA	98,41	100,00	99,18	99,22	100,00	99,59
14 LF-LA-RF	98,67	100,00	99,32	98,52	100,00	99,23
15 LF-LA-RF-RA	99,11	100,00	99,55	98,78	100,00	99,36

En el caso de la clasificación binaria de los ejercicios realizados, a la hora de elegir cuáles son las posiciones óptimas de las IMUs sobre las piernas o los brazos, es indiferente introducir la información de los ángulos de Euler a la hora de clasificar. Esto se debe a que como los resultados obtenidos al clasificar los ejercicios son bastante buenos, ya que todos los resultados de las métricas son superiores al 96 %, por lo que al introducir las características nuevas no conlleva que dichos resultados mejoren.

Se llega a las mismas conclusiones que utilizando solo los datos en crudo de las IMUs, en las que se concluye que las posiciones óptimas en las que situar las IMUs, tanto en los ejercicios de las piernas como en los brazos, son aquellas en las que se sitúan de forma cruzada o paralela. Dichas posiciones son 6 (LT-RT), 7 (LT-RS), 8 (LS-RT), 9 (LS-RS), en el caso de los ejercicios de pierna y 6 (LA-RA), 7 (LA-RF), 8 (LF-RA), 10 (LF-RF), en el caso de los de brazo.

4.2 Clasificación multiclase

Haciendo uso de clasificaciones multiclase se realizan las propuestas de clasificación para la identificación y la evaluación conjunta de los ejercicios. En la segunda propuesta (B) de clasificación expuesta en el apartado 3.5.1 se identifican los ejercicios de entre un catálogo previamente elegido. Y en la última propuesta de clasificación de los datos (C), se evalúan e identifican de forma conjunta los ejercicios. Los resultados obtenidos en dichas propuestas de clasificación se van a exponer en este apartado con el fin de obtener cuál de ellas es la que mejor se comporta ante los diferentes tipos de ejercicios y la calidad de ejecución de los mismos.

4.2.1 Identificación de los ejercicios

En este primer subapartado se exponen los resultados obtenidos al identificar los ejercicios mediante dos posibles formas. La primera de ellas es la identificación de los ejercicios bien hechos, representado como B1 en la Figura 4.5, y la segunda de ellas es la identificación de los ejercicios sin tener en cuenta que se estén realizando correcta o incorrectamente, representado como B2 en la Figura 4.5. El tipo de identificación B1 sólo necesita como entrada los datos bien hechos, en cambio la identificación B2 necesita tanto los

datos correctos e incorrectos de los ejercicios para identificarlos.

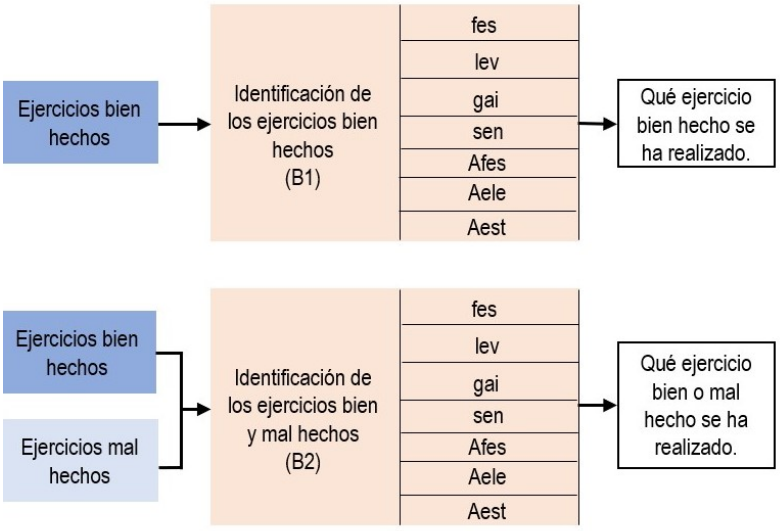


Figura 4.5: Entradas y salidas de los bloques de identificación de los ejercicios.

Los resultados obtenidos para identificar los ejercicios se detallan en la Tabla 4.20, en la que se refleja la media de los errores de clasificación obtenidos mediante una validación cruzada de 10 iteraciones con respecto a los tres algoritmos que han ofrecido mejores resultados en la evaluación de los ejercicios. Se han elegido los algoritmos de SVM_g, SVM_p y KNN ya que tanto el algoritmo de NB como el de NB_k, SVM, SVM_l y RF dan resultados con un porcentaje mayor de error que los elegidos.

Tabla 4.20: Errores de clasificación obtenidos al hacer la identificación de los ejercicios con la información de la aceleración lineal y la velocidad angular, introduciendo el número de datos referido al número de pares de datos para cada clasificación.

Error de identificación del ejercicio (%)		
Algoritmo	**Bien realizado**	**Bien y mal realizado**
SVM_g	0,11	1,49
SVM_p	0,06	1,52
KNN	0,02	1,43
Número de datos	2456	3360

Como se puede observar en la Tabla 4.20 los errores de la identificación de los ejercicios bien hechos están en torno a un 0 %. Esto se debe a que los ejercicios bien hechos están estipulados de una manera y son iguales para todo el mundo, lo que da lugar a que dicha identificación sea casi perfecta. En cambio, cuando se identifican tanto los ejercicios bien hechos como los mal hechos los errores aumentan en un 1,5 %. Este incremento en el error se debe a la existencia de los datos de los ejercicios mal hechos que producen más variabilidad en cuanto a la ejecución de dichos ejercicios. Aunque estos errores son mayores que los obtenidos al identificar sólo ejercicios bien hechos, son bastante buenos ya que se encuentran por debajo del 2 %. La diferencia de error entre SVM_g, SVM_p y KNN es inferior al 0,2 % para cada tipo de identificación, por lo que se puede afirmar que cualquiera de los tres es adecuado a la hora de identificar

qué ejercicio se está realizando.

Como el algoritmo de Machine Learning que mejor se comporta ante la evaluación de los ejercicios es KNN se evalúa la clasificación de ejercicios con variabilidad empleando dicho algoritmo. Asimismo, puesto que las posiciones óptimas de las IMUs son las cruzadas o paralelas, se elige la posición 8 (LS-RT ó LF-RA) para discutir cómo de óptima es la identificación de los ejercicios. Para ello se hace uso de la media de las matrices de confusión obtenidas al hacer una validación cruzada de 10 iteraciones de la identificación de los ejercicios bien hechos y mal hechos (Tabla 4.21). Se descarta la explicación de la identificación de los ejercicios bien hechos (B1) ya que su error ronda el 0 % y su matriz de confusión no tiene ningún tipo de error, es decir, es diagonal. Además, el segundo escenario de clasificación (B2) es el más realista, en el que la variabilidad de movimientos es mayor que únicamente realizando los ejercicios de forma correcta.

Tabla 4.21: Matriz de confusión para la identificación de los ejercicios con la clasificación B2. Los valores sombreados en verde son los clasificados correctamente y los rojos los que se han clasificado incorrectamente. Las filas representan las clases correctas, mientras que las columnas representan las clases predichas, señaladas con un ˆ.

	f$\hat{}$es	**sên**	**l$\hat{}$ev**	**g$\hat{}$ai**	**A$\hat{}$fes**	**A$\hat{}$ele**	**A$\hat{}$est**
fes	122	0	0	0	0	0	0
sen	0	116	0	1	0	0	0
lev	0	0	119	0	0	0	0
gai	0	0	0	113	0	0	0
Afes	0	0	0	0	118	0	0
Aele	0	0	0	0	2	122	0
Aest	0	0	0	0	0	0	123

En la Tabla 4.21 se representa el número de datos predichos correcta o incorrectamente a la hora de identificar qué ejercicio se está ejecutando. Como se puede observar, en la fila referida al ejercicio de sentadilla (sen) se han clasificado 116 datos correctamente, en cambio 1 dato se han clasificado como si fuese una caminar (gai). Esto se debe a que en los ejercicios mal hechos de marcha se flexionan las rodillas pudiendo parecer una sentadilla. Además, en la fila referida al ejercicio de elevación de brazos (Aele) se han clasificado 122 datos correctamente, mientras que 2 datos se han clasificado como flexo-extensión de codos (Afes). Esto explica que un ejercicio de Afes mal hecho y uno de Aele mal hecho pueden ser muy similares entre sí.

La clasificación realizada con las posiciones de las IMUs cruzadas sobre las piernas o los brazos es bastante buena ya que, aunque haya diferentes clasificaciones erróneas éstas son muy bajas y son casi imperceptibles con respecto a las clasificaciones bien realizadas. Es por ello que se puede concluir que el mejor algoritmo y las posiciones óptimas elegidas en la parte de la evaluación de los ejercicios son muy buenas a la hora de identificar los mismos.

Para comprobar los resultados obtenidos con la matriz de confusión se muestra en la Tabla 4.22 las métricas obtenidas cuando se hace uso de dos IMUs, en los ejercicios de pierna y los de brazo.

Si se comparan las Tablas representadas en la Tabla 4.22 con las obtenidas en el apartado 4.1.1 se puede

Tabla 4.22: Especificidad, sensibilidad y exactitud de las posiciones en las que se sitúan dos IMUs tanto para los ejercicios de pierna como para los de brazo.

(%) Ejercicios de pierna				(%) Ejercicios de brazo			
Posición	**E**	**S**	**Ex**	**Posición**	**E**	**S**	**Ex**
5 LS-LT	99,80	99,24	99,72	**5 LF-LA**	99,92	99,44	99,84
6 LT-RT	99,83	99,33	99,76	**6 LA-RA**	99,46	98,69	99,35
7 LT-RS	99,82	99,66	99,80	**7 LA-RF**	99,86	99,03	99,74
8 LS-RT	99,79	99,58	99,76	**8 LF-RA**	99,90	99,52	99,84
9 LS-RS	99,78	99,49	99,74	**9 LF-RF**	99,92	99,52	99,86
10 RS-RT	99,51	99,48	99,51	**10 RF-RA**	99,93	99,04	99,80

observar que la diferencia entre las métricas al utilizar dos IMUs para evaluar e identificar los ejercicios varían muy poco, ya que se obtienen, en ambos casos, especificidades, sensibilidades y exactitudes que rondan entre el 98 y el 99 %. Esto ayuda a corroborar lo discutido en la matriz de confusión ya que la discusión de las mejores posiciones utilizadas en la evaluación de los ejercicios es totalmente válida en la identificación de los mismos.

4.2.2 Identificación y evaluación de los ejercicios

En este subapartado se realiza la comparación entre dos tipos de identificación de los ejercicios y evaluación de los mismos. El primer tipo de identificación y evaluación de ejercicios, representado como C1 en la Figura 4.6, es una clasificación en la que se hace uso de 8 clases, de las cuales 7 se encargan de identificar el ejercicio realizado correctamente y la otra clase evalúa si alguno de ellos se realiza mal. El segundo tipo de identificación y evaluación de ejercicios, representado como C2 en la Figura 4.6, es aquella en la que se hace uso de 14 clases que representan qué ejercicio es el realizado y la evaluación del mismo.

Habiendo demostrado que el mejor algoritmo para la aplicación es KNN y las posiciones óptimas son las cruzadas o paralelas, se hace uso de dicho algoritmo y de la posición 8 a la hora de representar las matrices de confusión obtenidas al hacer una validación cruzada de 10 iteraciones.

En la Tabla 4.23 se representa la matriz de confusión de la clasificación de 8 clases donde se determina qué ejercicio bien hecho se está realizando y si se realiza mal algún ejercicio (C1). Las columnas representan las clases predichas y las filas representan las clases correctas.

Como se puede observar en dicha matriz de confusión, la clasificación de las clases es muy buena ya que del total de datos sólo 3 son mal predichos. Los tres datos que se han clasificado mal son los detectados como caminar (gai) que en realidad son ejercicios mal hechos que deberían estar dentro de la clase *wrng*. Este error de clasificación se puede dar debido a que dentro de la clase *wrng* hay muchos ejercicios aleatorizados con características diferentes lo que puede dar lugar a que se confundan datos debido a tal variabilidad.

La siguiente clasificación propuesta es en la que se hace uso de 14 clases a partir de las cuales se evalúa el ejercicio realizado y la calidad del mismo (C2). En la Tabla 4.24 se representa la matriz de confusión para dicha clasificación.

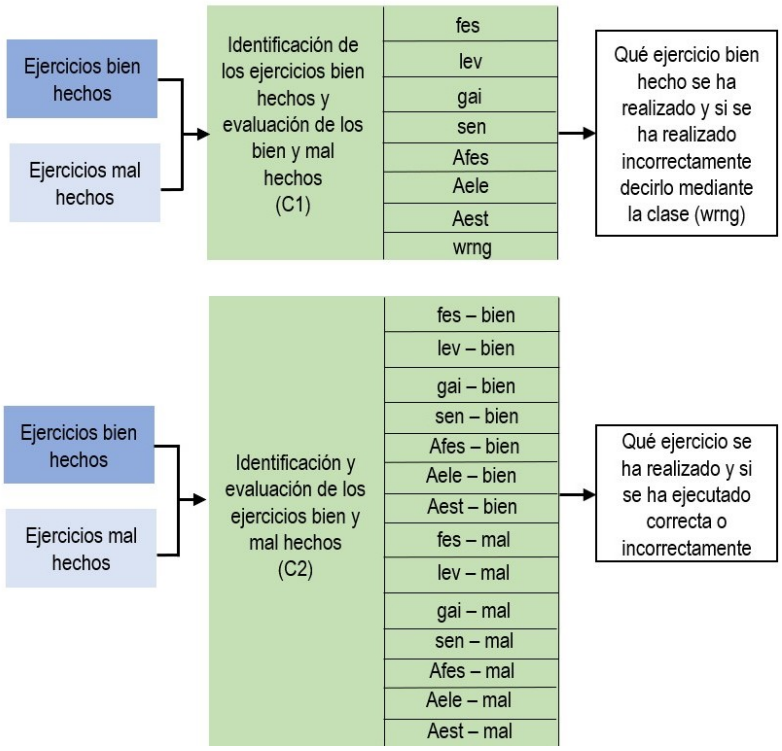


Figura 4.6: Entradas y salidas del bloque de evaluación e identificación de los ejercicios.

Tabla 4.23: Matriz de confusión para la identificación y la evaluación de los ejercicios tanto de pierna como de brazo (clasificación C1). Siendo "*wrng*" la clase en la que se aleatorizan todos los ejercicios mal hechos. Los valores sombreados en verde son los clasificados correctamente y los rojos los que se han clasificado incorrectamente.

	f̂es	**sên**	**l̂ev**	**ĝai**	**Âfes**	**Âele**	**Âest**	**wr̂ng**
fes	89	0	0	0	0	0	0	0
sen	0	87	0	0	0	0	0	0
lev	0	0	91	0	0	0	0	0
gai	0	0	0	87	0	0	0	0
Afes	0	0	0	0	88	0	0	0
Aele	0	0	0	0	0	84	0	0
Aest	0	0	0	0	0	0	90	0
wrng	0	0	0	3	0	0	0	81

En la Tabla 4.24 se puede observar que hay más cantidad de datos mal predichos que en la Tabla 4.23. Esto se debe a que existe un mayor número de clases lo que da lugar a un incremento del error de clasificación. En el ejercicio de elevación de brazos bien hecho (Aele0) se han predicho incorrectamente 2 datos que en realidad son de flexo-extensión de codos bien (Afes0) y mal hecho (Afes1). Esto se debe a que la diferencia que existe entre el ejercicio Afes y Aele es muy poca si el ejercicio de Afes se realiza mal. Tal y como se detalla en la Tabla 4.24, se predicen mejor los ejercicios de brazos que los de pierna ya que éstos últimos detectan erróneamente más número de ejercicios que los de brazos. Aunque se detecten más errores que en la Tabla 4.23, los resultados siguen siendo buenos ya que se obtienen 29 datos mal clasificados en un total del 1033 datos, es decir, resulta en un $2,8\,\%$ de error. Esto implica que el número

Tabla 4.24: Matriz de confusión para la identificación y la evaluación de los ejercicios tanto de pierna como de brazo (clasificación C2). Siendo "0" la repetición del ejercicio bien hecha y "1" la repetición mal hecha.

	fês0	**fês1**	**sên0**	**sên1**	**lêv0**	**lêv1**	**gâi0**	**gâi1**	**Afês0**	**Afês1**	**Aêle0**	**Aêle1**	**Aêst0**	**Aêst1**
fes0	87	0	0	0	0	0	0	0	0	0	0	0	0	0
fes1	0	88	0	0	0	0	0	0	1	0	0	0	0	0
sen0	0	0	90	0	0	0	0	0	0	1	0	0	0	0
sen1	0	0	0	84	0	0	0	0	0	0	0	0	0	0
lev0	0	0	0	0	93	0	0	0	0	0	0	0	0	0
lev1	0	0	0	0	0	84	0	0	0	0	0	0	0	0
gai0	0	0	0	0	0	0	89	0	0	0	0	0	0	0
gai1	3	0	0	0	0	0	0	56	0	0	0	0	0	0
Afes0	0	6	0	0	0	0	0	0	51	1	1	0	0	0
Afes1	0	0	5	0	0	0	0	0	0	51	1	0	0	0
Aele0	0	0	0	2	0	0	0	0	0	0	56	0	0	0
Aele1	0	0	0	0	2	0	0	0	0	0	0	55	0	0
Aest0	0	0	0	0	0	3	0	0	0	0	0	1	57	0
Aest1	0	0	0	0	0	0	2	0	0	0	0	0	0	63

de predicciones incorrectas con respecto al total de predicciones es mínimo.

Es por ello que se puede concluir que el tipo de clasificación que detecta peor los ejercicios y, por consiguiente, tiene más error de clasificación es el que hace uso de 14 clases, mientras que la clasificación en la que se hace uso de 8 clases ofrece un número menor de error de clasificación. Sin embargo, esta clasificación es menos específica, ya que al realizar mal un ejercicio no se identifica cuál es el ejercicio a realizar. Por tanto, si se quiere obtener mejor resultado a costa de perder información se elige la clasificación con 8 clases. En cambio si se quiere detectar también los ejercicios mal hechos se hace uso de la clasificación con 14 clases en la que se ofrece un mayor error pero discierne entre las clases mal hechas con un alto porcentaje de aciertos.

4.2.3 Identificación y evaluación de los ejercicios con información de los ángulos de Euler

Tal y como se ha expuesto en la introducción de este capítulo, el fin de introducir información de los ángulos de Euler que sufren las IMUs es poder mejorar los resultados obtenidos en la clasificación sin hacer uso de los mismos. Como se ha detallado en los apartados 4.2.1 y 4.2.2, los resultados obtenidos son muy precisos ya que, para las posiciones óptimas elegidas tanto en los ejercicios de la pierna como en los de brazo, tienen un porcentaje de error muy pequeño y una precisión cerca del 100 %. Estos resultados sugieren que lo más óptimo es no utilizar la información de los ángulos de Euler ya que implica más procesamiento y la mejora en la aplicación, en caso de haberla, es mínima.

Capítulo 5

Conclusiones y trabajos futuros

5.1 Conclusiones

El aumento de la esperanza de vida en los países desarrollados ha dado lugar a un interés creciente por alcanzar un envejecimiento saludable, el cual implica comida sana y ejercicio diario. Actualmente hay multitud de dispositivos y aplicaciones que permiten monitorizar la actividad a través de la estimación del número de pasos utilizando una IMU. Hay algunos sistemas que incluso discriminan entre si estás de pie, sentado o tumbado. Sin embargo, no hay muchos trabajos que aborden la monitorización de ejercicios como ayuda al paciente, haciendo las veces de un entrenador virtual, y a los cuidadores o facultativos que pueden evaluar el desempeño del los ejercicios físicos, los cuales pueden estar pautados de forma específica dentro algún programa concreto de fragilidad, prevención de caídas, etc.

Este Trabajo de Fin de Grado se ha realizado motivado por esta necesidad del seguimiento de rutinas en el domicilio, permitiendo monitorizar los ejercicios que realiza el paciente, mediante la obtención y el posterior tratamiento de los datos ofrecidos por las IMUs.

En este estudio se han realizado diferentes clasificaciones complejas utilizando métodos de Machine Learning. El fin de dichas clasificaciones es identificar qué ejercicio se está realizando así como la evaluación del mismo, es decir, si se realiza correcta o incorrectamente. Para las distintas clasificaciones se determina el algoritmo que mejores resultados obtiene así como las mejores posiciones de las IMUs para esta monitorización de ejercicios. Para ello se ha trabajado con un catálogo de siete ejercicios diferentes donde tres trabajan las extremidades inferiores, tres las superiores y el último ejercicio era el de marcha. Además, se ha hecho uso de cuatro IMUs para los ejercicios del tren inferior, con dos en cada miembro, y el equivalente en el tren superior. La entrada a los modelos de Machine Learning han sido cuatro características obtenidas de las señales en crudo obtenidas por las IMUs, es decir, la velocidad angular y aceleración lineal, aunque también se ha analizado el efecto sobre la clasificación al considerar la estimación de los ángulos de Euler como entrada adicional. Las características empleadas son la medida, desviación estándar, máximo y mínimo de los segmentos en los que se han dividido mediante una ventana deslizante las señales evaluadas.

Se ha desarrollado una clasificación binaria para evaluar un ejercicio concreto en bien o mal realizado

obteniendo sensibilidades del orden de 99, 50 % con el método KNN, que además ha demostrado ser uno de los más óptimos para este estudio. Además se ha desarrollado una clasificación multiclase que ha permitido identificar y evaluar los ejercicios de entre un catálogo propuesto obteniendo exactitudes del orden de 99, 80 % con el método KNN.

Además se ha estudiado la optimización del número y posiciones de las IMUs y se ha concluido que la mejor opción es hacer uso de dos unidades de medida tanto en los ejercicios de brazos como en los de pierna. Dichas IMUs tendrán que estar situadas en posiciones cruzadas o paralelas de cada extremidad inferior o superior.

Se han analizado distintos planteamientos tanto de evaluación como de identificación de ejercicios y se concluye que los errores son inferiores al 2 % haciendo uso del algoritmo KNN y de dos IMUs. Por lo tanto, se puede concluir que cumpliendo con los objetivos marcados se ha desarrollado un clasificador que a partir de todos los datos, todos los ejercicios bien y mal hechos, es capaz de identificar cuál se está realizando y, además, evaluar si se ha realizado bien o no con una sensibilidad del 99 % y una especificidad del 98 %.

Los resultados obtenidos son muy buenos, lo que hace prometedora la propuesta pero será necesario evaluar el sistema de una forma que se asemeje de otra forma a la aplicación buscada. Esto es, evaluando los resultados en un sujeto, desconocido para los clasificadores, mediante una validación cruzada de leave-one-out.

Por último, se ha estudiado la posibilidad de mejorar los clasificadores basados en Machine Learning aumentando el número de características a la entrada y para ello se han incluido los ángulos del movimiento. Dichos ángulos se han representado a través de los ángulos de Euler que se han estimado a partir de los datos de una IMU, usando un UKF. Para el conjunto de datos analizados los resultados admiten poco margen de mejora, lo que hace que el aumento de características no implique una mejora significativa de parámetros estadísticos de métrica. Los datos varían en menos de un 1 % en la mayoría de los casos, tanto respecto a la mejora como al empeoramiento, por lo que no se puede considerar una variación significativa. La causa principal de esta variación es la nueva aleatorización de datos para la validación cruzada aleatoria que se hace al emplear los ángulos de Euler, de forma que no se parte de los mismos datos al evaluar únicamente los datos de las IMUs que al introducir además los ángulos del movimiento.

Asimismo, debido a la pequeña magnitud de la variación de los resultados y los buenos resultados al no emplear los ángulos de Euler, se concluye que a la hora de clasificar con los datos empleados en este trabajo lo más óptimo no es utilizar la información relativa a los ángulos de Euler ya que requiere un mayor procesamiento de datos y la mejora es mínima.

5.2 Trabajos futuros

Con la elaboración de este Trabajo de Fin de Grado se plantean una serie de ideas o mejoras que pueden ser propuestas como líneas de trabajos futuros:

Validación de nuevos sujetos. Una primera propuesta de línea futura es la evaluación de los algoritmos en una nueva persona, es decir, empleando el método de remuestreo de *leave one out*. Esto es, se entrenan con los datos obtenidos hasta el momento, pero a la hora de testear el modelo se introducen los datos de dicha nueva persona. Se puede pronosticar que los errores de clasificación podrían aumentar realizando este método de remuestreo, dejando un margen de mejora para el estudio de los ángulos de Euler.

Evaluación de algoritmos de Machine Learning con un mayor número de características de entrada. Una segunda propuesta de línea futura es la evaluación de los algoritmos introduciendo un mayor número de características de entrada, como la varianza, el rango, los coeficientes de correlación de Pearson. También se quiere introducir características del dominio de frecuencia como la componente de frecuencia máxima, la componente de frecuencia media, la densidad espectral de energía y entropía.

Creación de una aplicación para la evaluación e identificación de los ejercicios. La línea futura final es la creación de una aplicación que ofrezca los resultados de la evaluación e identificación de los ejercicios en tiempo real. De esta forma, el usuario podrá ser capaz de saber la evaluación e identificación de los ejercicios que realiza simplemente mirando una aplicación instalada en su smartphone.

Capítulo 6

Presupuesto

En este capítulo se realiza una estimación de los costes del proyecto, que incluyen tanto los materiales empleados como la mano de obra necesaria para realizar el mismo. El coste del material se divide en dos partes: materiales físicos detallados en la Tabla 6.1 y software representado en la Tabla 6.2.

Tabla 6.1: Coste del material físico.

COSTE DEL MATERIAL FÍSICO				
Dispositivo	**Uso (en meses)**	**Duración (en años)**	**Precio**	**Coste**
Portátil Lenovo IdeaPad	5	4	449,00 €	46,77 €
4 IMUs	5	3	1.432,00 €	198,89 €
Cable USB IMU	5	4	5,65 €	0,59 €
Total coste material físico				**246,25 €**

Tabla 6.2: Coste del software.

COSTE DEL SOFTWARE				
Programa / licencia	**Uso (en meses)**	**Duración (en años)**	**Precio**	**Coste**
MATLAB 2020a	5	4	449,00 €	46,77 €
Licencia MATLAB	5	1	167,76 €	69,90 €
Microsoft Office 365	5	3	412,50 €	57,29 €
Total coste software				**173,96 €**

Por lo tanto, el coste total del material es la suma de los costes de los materiales físicos y el software, como puede observarse en la Tabla 6.3.

Tabla 6.3: Coste total del material.

TOTAL COSTE MATERIAL	**420,21 €**
Total coste material físico	**246,25 €**
Total coste materiales	**173,96 €**

Una vez estimados los costes del material físico, se calculan los costes de mano de obra a partir de las horas realizadas tanto en la grabación de datos como en la redacción del proyecto (Tabla 6.4).

Tabla 6.4: Coste de mano de obra.

COSTE DE MANO DE OBRA			
Descripción	**Coste/hora (€)**	**Horas**	**Coste**
Trabajo de ingeniería	30,00 €	450	13.500,00 €
Grabación de datos	25,00 €	275	6.875,00 €
Redacción del proyecto	20,00 €	200	4.000,00 €
Total coste mano de obra			**24.375,00 €**

El coste total del proyecto es la suma de los costes de material físico y mano de obra, como se detalla en la Tabla 6.5.

Tabla 6.5: Coste total del proyecto

Total coste realización del proyecto	**24.795,21 €**
Total coste material	**420,21 €**
Total coste mano de obra	**24.375,00 €**

Para realizar el presupuesto de ejecución, se añade al coste total del proyecto el beneficio de ejecución del mismo, estimado, en este caso, en el 15 %, según se detalla en la Tabla 6.6.

Tabla 6.6: Presupuesto de ejecución.

Presupuesto de ejecución	**28.514,49 €**
Total coste realización del proyecto	**24.795,21 €**
Beneficio del proyecto (15 %)	**3.719,28 €**

Por último, para obtener el presupuesto final del proyecto hay que añadir al presupuesto de ejecución los honorarios de realización del proyecto (7 %) obteniendo el importe de la base imponible del presupuesto y calcular, a continuación, el IVA del 21 %.

Tabla 6.7: Presupuesto final del proyecto.

Presupuesto final del proyecto	**36.917,71 €**
Presupuesto de ejecución	**28.514,49 €**
Honorarios profesionales (7 %)	**1.996,01 €**
Base imponible del presupuesto	**30.510,51 €**
IVA (21 %)	**6.407,21 €**

El presupuesto total del proyecto asciende a la cifra de TREINTA Y SEIS MIL NOVECIENTOS DIECISIETE EUROS CON SETENTA Y UN CÉNTIMOS.

Pliego de condiciones

En este capítulo se va a explicar el Pliego de Condiciones que se debe cumplir a la hora de desarrollar el proyecto. El objetivo principal de este proyecto es identificar ejercicios y evaluar su calidad mediante la información suministrada por los sensores inerciales y la clasificación de los mismos usando algoritmos de Machine Learning. Para ello es necesario comprender la matemática asociada a los sensores inerciales así como ser capaces de analizar el funcionamiento de dichos sensores a la hora de utilizarlos en aplicaciones relacionadas con la rehabilitación. Como se ha explicado en el apartado 3.3, el proyecto debe ser evaluado y aprobado por un comité de ética y, además, los voluntarios deben dar el consentimiento informado para poder tratar sus datos.

Requerimientos a nivel hardware

Para la puesta en marcha de este proyecto se requiere el uso de las **NGIMUs** [12] de x-io Technologies, las cuales deben estar situadas en el muslo y la espinilla de ambas piernas a la hora de realizar los ejercicios de pierna, y en los brazos y antebrazos en la ejecución de los ejercicios de brazo. Su diseño compacto junto con la capacidad de almacenar información en una tarjeta SD y una batería de larga duración hace que sea un dispositivo de fácil uso para las aplicaciones enfocadas a la clasificación de la rehabilitación. En este proyecto, las IMUs se han utilizado en el capítulo 3 a la hora de grabar los movimientos realizados por los voluntarios. Sus principales características técnicas son:

- Sensores y adquisición de datos:
  - Giróscopo triaixial ($\pm 2000\,^{o}$/s, $F_s = 400\,$Hz).
  - Acelerómetro triaixial ($\pm 16\,$g, $F_s = 400\,$Hz).
  - Magnetómetro triaixial ($\pm 1300\,\mu$T).
  - Presión, humedad y temperatura.
- Procesamiento de datos internos:
  - Calibración de sensores.
  - Algoritmos AHRS capaces de proporcionar una mejor precisión y fiabilidad de la orientación de un dispositivo. Para ello hace uso de cuaterniones, ángulos de Euler, matrices de rotación, aceleración lineal y velocidad angular.
- Interfaces de comunicación:

- USB.
- Puerto serie.
- Tarjeta micro SD.

Se requiere, además, de un **ordenador portátil**, para realizar las pruebas de campo con los voluntarios, capaz de instalar Matlab para ejecutar los programas creados en dicho entorno. Por último, a lo largo del desarrollo del TFG se han utilizado distintas herramientas de trabajo como cintas ajustables para portar las IMUs.

Requerimientos a nivel software

Se requiere el uso de diversas herramientas software para la puesta en marcha de este proyecto:

- **NGIMU GUI v1.14:** software propiedad de x-io Technologies el cual sirve como interfaz para acceder a los datos proporcionados por las IMUs y para configurar los parámetros de funcionamiento como puede ser la frecuencia de muestreo.
- **NGIMU SD Card File Converter:** software propiedad de x-io Technologies el cual sirve para tratar los datos de las IMUs y transformar los archivos *".XIO"* en archivos *".csv"*, tal y como se explica en el anexo A.
- **Matlab R2020a:** Se trata de la versión de Matlab instalada y usada a la hora de la realización del proyecto. La licencia obtenida es la de estudiante proporcionada por la Universidad de Alcalá de Henares.
- **Microsoft Office Pro Plus:** software utilizado para la elaboración de Tablas (Microsoft Excel) y para la creación de diagramas y gráficos (Microsoft PowerPoint).
- **Overleaf - LaTeX:** software utilizado para la elaboración de la memoria del proyecto.

Comité de Ética

Los participantes de este estudio fueron reclutados por el grupo de investigación GEINTRA de la Universidad de Alcalá. En este estudio participaron 14 voluntarios sanos y robustos con edades comprendidas entre los 20 y 50 años. El estudio se llevó a cabo al amparo del proyecto FrailCheck (SBPLY/17/180501/000392) en el laboratorio MoCap de la UAH, siguiendo todos los protocolos COVID-19 exigidos por la Universidad. El Hospital Universitario de Guadalajara aprobó el protocolo del presente estudio como una adenda al protocolo que ya se tenía aprobado para el proyecto FrailCheck (Institutional Review Board No.2018.22.PR, protocol version V.1. dated 21/12/2020). Todos los participantes lo hicieron de forma voluntaria como quedó recogido en el consentimiento informado que firmaron previamente a la ejecución de los ejercicios.

Anexos

A Generación de la base de datos

Para renombrar los archivos de cada IMU de una manera factible, se define el formato de nombrado de los archivos de todas las personas voluntarias que iban a realizar los ejercicios.

Estos archivos siguen siempre un mismo formato, el cual viene detallado a continuación:

1. Índice del sujeto diario, el cual señala si ha sido la primera persona en realizar los ejercicios (1), la segunda (2), etc, de cada día.
2. Ejercicio realizado relativo al archivo.
3. Si el ejercicio está bien hecho (0) y si el ejercicio está mal hecho (1).
4. La serie de cada ejercicio (1 ó 2)

Para explicar con exactitud el proceso de organización y renombre de datos se supone que una persona, la segunda de ese día, ha realizado los ejercicios de brazos y piernas el 29 de Noviembre de 2020. Sus datos se guardarán tal y como aparece en la Figura A.1

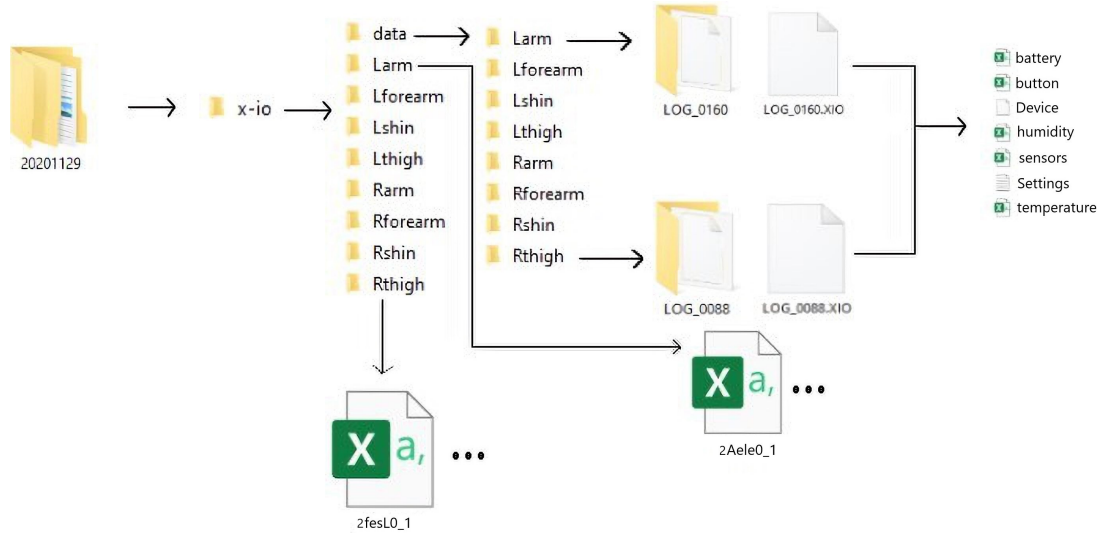


Figura A.1: Volcado de datos de las IMUs.

La organización de los datos por fecha sigue el formato mostrado en la Figura A.1. Como se puede observar,

los datos se guardan dentro de carpetas que se nombran con el año, mes y día en el que se realizaron los ejercicios. Dentro de esa carpeta se encuentra una carpeta denominada "\x-io", ya que usamos las IMUs de X-IO Technologies. En esta carpeta hay nueve subcarpetas. Una de ellas es "\data" y las demás son las relativas a las posiciones de las IMUs tanto en brazos como en piernas. Dentro de la carpeta "\data" hay ocho subcarpetas con los datos obtenidos de las IMUs con su extensión "*.XIO*" y los datos convertidos a partir del software de transformación de datos. Tal y como se puede ver en la Figura A.1, dentro de las carpetas "\LOG_0160" y "\LOG_0088" se encuentra tanto el archivo *sensors.csv* que se va a usar como los demás que se han determinado anteriormente. Por último, en las subcarpetas que hay en el directorio "\x-io", a excepción de "\data", están los archivos renombrados relativos a cada IMU y a cada ejercicio.

En la Figura A.1 se puede observar que dentro de la carpeta "\Larm" hay un archivo denominado *2Aele0_1*, que se refiere a que este segundo sujeto (2) ha realizado el ejercicio de elevación de brazos (Aele) bien hecho (0) de elevación de brazos y es su primera serie (1).

B Algoritmos de Machine Learning utilizados para la evaluación de los ejercicios

```
function [err_NB_Train,err_NB_Test,err_NB_Train_kernel,err_NB_Test_kernel,...
    err_SVM_Train,err_SVM_Test,err_SVM_lin_Train,err_SVM_lin_Test,...
    err_SVM_gauss_Train,err_SVM_gauss_Test,err_SVM_pol_Train,...
    err_SVM_pol_Test,err_RF_Train,err_RF_Test,err_KNN_Train,...
    err_KNN_Test,CMNB,orderNB,CMNB_k,orderNB_k,CMSVM,orderSVM,CMSVMl,...
    orderSVMl,CMSVMg,orderSVMg,CMSVMp,orderSVMp,CMRF,orderRF,CMKNN,...
    orderKNN]=clasificadores_bin(XTrain, YTrain, XTest, YTest)


    % Naive Bayes

    mdl_NB_1=fitcnb(XTrain, YTrain);
    pred_Xtrain_NB_1=predict(mdl_NB_1,XTrain);
    err_NB_Train=loss(mdl_NB_1, XTrain, YTrain);
    pred_Xtest_NB_1=predict(mdl_NB_1,XTest);
    err_NB_Test=loss(mdl_NB_1, XTest, YTest);
    [CMNB,orderNB] = confusionmat(YTest,pred_Xtest_NB_1);

    % NB con kernel
    mdl_NB_2=fitcnb(XTrain, YTrain, "DistributionNames", "kernel");
    pred_Xtrain_NB_2=predict(mdl_NB_2,XTrain);
    err_NB_Train_kernel=loss(mdl_NB_2, XTrain, YTrain);
    pred_Xtest_NB_2=predict(mdl_NB_2,XTest);
    err_NB_Test_kernel=loss(mdl_NB_2, XTest, YTest);
    [CMNB_k,orderNB_k] = confusionmat(YTest,pred_Xtest_NB_2);
```

```
% SVM

mdlSVM = fitcsvm(XTrain, YTrain,'Standardize',true);
pred_Xtrain_SVM=predict(mdlSVM,XTrain);
err_SVM_Train=loss(mdlSVM, XTrain, YTrain);
pred_Xtest_SVM=predict(mdlSVM,XTest);
err_SVM_Test=loss(mdlSVM,XTest, YTest);
[CMSVM,orderSVM] = confusionmat(YTest,pred_Xtest_SVM);

% Kernel Lineal
mdlSVM_lin = fitcsvm(XTrain, YTrain,'KernelFunction','linear',...
    'KernelScale','auto','Standardize',true);
pred_Xtrain_SVM_lin=predict(mdlSVM_lin, XTrain);
err_SVM_lin_Train=loss(mdlSVM_lin,XTrain, YTrain);
pred_Xtest_SVM_lin=predict(mdlSVM_lin ,XTest);
err_SVM_lin_Test=loss(mdlSVM_lin, XTest, YTest);
[CMSVMl,orderSVMl] = confusionmat(YTest,pred_Xtest_SVM_lin);

% Kernel Gaussiana
mdlSVM_gauss = fitcsvm(XTrain, YTrain,'KernelFunction','gaussian',...
    'KernelScale','auto','Standardize',true);
pred_Xtrain_SVM_gauss=predict(mdlSVM_gauss,XTrain);
err_SVM_gauss_Train=loss(mdlSVM_gauss,XTrain, YTrain);
pred_Xtest_SVM_gauss=predict(mdlSVM_gauss,XTest);
err_SVM_gauss_Test=loss(mdlSVM_gauss,XTest, YTest);
[CMSVMg,orderSVMg] = confusionmat(YTest,pred_Xtest_SVM_gauss);

% Kernel Polinomial
mdlSVM_pol = fitcsvm(XTrain, YTrain,'KernelFunction','polynomial',...
    'KernelScale','auto','Standardize',true);
pred_Xtrain_SVM_pol=predict(mdlSVM_pol,XTrain);
err_SVM_pol_Train=loss(mdlSVM_pol, XTrain, YTrain);
pred_Xtest_SVM_pol=predict(mdlSVM_pol, XTest);
err_SVM_pol_Test=loss(mdlSVM_pol, XTest, YTest);
[CMSVMp,orderSVMp] = confusionmat(YTest,pred_Xtest_SVM_pol);


% Random Forests

mdlRF = fitcensemble(XTrain, YTrain, "method", "Bag");
pred_Xtrain_RF=predict(mdlRF,XTrain);
err_RF_Train=loss(mdlRF,XTrain, YTrain);
pred_Xtest_RF=predict(mdlRF,XTest);
err_RF_Test=loss(mdlRF, XTest, YTest);
[CMRF,orderRF] = confusionmat(YTest,pred_Xtest_RF);
```

```
    % KNN

    mdlKNN = fitcknn(XTrain, YTrain,'Standardize',true);
    pred_Xtrain_KNN=predict(mdlKNN,XTrain);
    err_KNN_Train=loss(mdlKNN, XTrain, YTrain);
    pred_Xtest_KNN=predict(mdlKNN,XTest);
    err_KNN_Test=loss(mdlKNN,XTest, YTest);
    [CMKNN,orderKNN] = confusionmat(YTest,pred_Xtest_KNN);



end
```

C Algoritmos de Machine Learning utilizados para la evaluación e identificación de los ejercicios

```
function[err_NB_Train,err_NB_Test,err_NB_Train_kernel,err_NB_Test_kernel,...
    err_SVM_Train,err_SVM_Test,err_SVM_lin_Train,err_SVM_lin_Test,...
    err_SVM_gauss_Train,err_SVM_gauss_Test,err_SVM_pol_Train,...
    err_SVM_pol_Test,err_RF_Train,err_RF_Test,err_KNN_Train,...
    err_KNN_Test,CMNB,orderNB,CMNB_k,orderNB_k,CMSVM,orderSVM,CMSVMl,...
    orderSVMl,CMSVMg,orderSVMg,CMSVMp,orderSVMp,CMRF,orderRF,CMKNN,...
    orderKNN]=clasificadores(XTrain, YTrain, XTest, YTest)


    % Naive Bayes

    mdl_NB_1=fitcnb(XTrain, YTrain);
    pred_Xtrain_NB_1=predict(mdl_NB_1,XTrain);
    err_NB_Train=loss(mdl_NB_1, XTrain, YTrain);
    pred_Xtest_NB_1=predict(mdl_NB_1,XTest);
    err_NB_Test=loss(mdl_NB_1, XTest, YTest);
    [CMNB,orderNB] = confusionmat(YTest,pred_Xtest_NB_1);

    % NB con kernel
    mdl_NB_2=fitcnb(XTrain, YTrain, "DistributionNames", "kernel");
    pred_Xtrain_NB_2=predict(mdl_NB_2,XTrain);
    err_NB_Train_kernel=loss(mdl_NB_2, XTrain, YTrain);
    pred_Xtest_NB_2=predict(mdl_NB_2,XTest);
    err_NB_Test_kernel=loss(mdl_NB_2, XTest, YTest);
    [CMNB_k,orderNB_k] = confusionmat(YTest,pred_Xtest_NB_2);
```

```
% SVM

template = templateSVM('Standardize',true);
mdlSVM = fitcecoc(XTrain, YTrain, 'Learners', template,...
    'FitPosterior',true);
pred_Xtrain_SVM=predict(mdlSVM,XTrain);
err_SVM_Train=loss(mdlSVM, XTrain, YTrain);
pred_Xtest_SVM=predict(mdlSVM,XTest);
err_SVM_Test=loss(mdlSVM,XTest, YTest);
[CMSVM,orderSVM] = confusionmat(YTest,pred_Xtest_SVM);

% Kernel Lineal
template_lin = templateSVM('Standardize',true,'KernelFunction',...
    'linear','KernelScale','auto');
mdlSVM_lin = fitcecoc(XTrain, YTrain, "Learners", template_lin,...
    'FitPosterior',true);
pred_Xtrain_SVM_lin=predict(mdlSVM_lin, XTrain);
err_SVM_lin_Train=loss(mdlSVM_lin,XTrain, YTrain);
pred_Xtest_SVM_lin=predict(mdlSVM_lin ,XTest);
err_SVM_lin_Test=loss(mdlSVM_lin, XTest, YTest);
[CMSVMl,orderSVMl] = confusionmat(YTest,pred_Xtest_SVM_lin);

% Kernel Gaussiana
template_gauss = templateSVM('Standardize',true,'KernelFunction',...
    'gaussian','KernelScale','auto');
mdlSVM_gauss = fitcecoc(XTrain, YTrain, "Learners", template_gauss,...
    'FitPosterior',true);
pred_Xtrain_SVM_gauss=predict(mdlSVM_gauss,XTrain);
err_SVM_gauss_Train=loss(mdlSVM_gauss,XTrain, YTrain);
pred_Xtest_SVM_gauss=predict(mdlSVM_gauss,XTest);
err_SVM_gauss_Test=loss(mdlSVM_gauss,XTest, YTest);
[CMSVMg,orderSVMg] = confusionmat(YTest,pred_Xtest_SVM_gauss);


% Kernel Polinomial
template_pol = templateSVM('Standardize',true,'KernelFunction',...
    'polynomial','KernelScale','auto');
mdlSVM_pol = fitcecoc(XTrain, YTrain, "Learners", template_pol,...
    'FitPosterior',true);
pred_Xtrain_SVM_pol=predict(mdlSVM_pol,XTrain);
err_SVM_pol_Train=loss(mdlSVM_pol, XTrain, YTrain);
pred_Xtest_SVM_pol=predict(mdlSVM_pol, XTest);
err_SVM_pol_Test=loss(mdlSVM_pol, XTest, YTest);
```

```
    [CMSVMp, orderSVMp] = confusionmat(YTest, pred_Xtest_SVM_pol);


    % Random Forests

    t = templateTree('Reproducible',true);
    mdlRF = fitcensemble(XTrain, YTrain, 'Method','Bag',...
        'NumLearningCycles',200,'Learners',t);
    pred_Xtrain_RF=predict(mdlRF, XTrain);
    err_RF_Train=loss(mdlRF, XTrain, YTrain);
    pred_Xtest_RF=predict(mdlRF, XTest);
    err_RF_Test=loss(mdlRF, XTest, YTest);
    [CMRF, orderRF] = confusionmat(YTest, pred_Xtest_RF);


    % KNN

    template_KNN = templateKNN('Standardize',true);
    mdlKNN = fitcecoc(XTrain, YTrain, 'Learners', template_KNN,...
        'FitPosterior',true);
    pred_Xtrain_KNN=predict(mdlKNN, XTrain);
    err_KNN_Train=loss(mdlKNN, XTrain, YTrain);
    pred_Xtest_KNN=predict(mdlKNN, XTest);
    err_KNN_Test=loss(mdlKNN, XTest, YTest);
    [CMKNN, orderKNN] = confusionmat(YTest, pred_Xtest_KNN);

end
```

D Código para la evaluación de ejercicios

```
imuRS = [1:12 25:36];
imuRT = [13:24 37:48];
imuLS = [49:60 73:84];
imuLT = [61:72 85:96];


evalIMUposition={imuLT, imuLS, imuRT, imuRS, [imuLS imuLT], [imuLT imuRT],...
    [imuLT imuRS], [imuLS imuRT], [imuLS imuRS], [imuRS imuRT],...
    [imuLS imuLT imuRT], [imuLS imuRS imuRT], [imuLT imuRS imuRT],...
    [imuLS imuLT imuRS], [imuLS imuLT imuRS imuRT]};

exercises_0_1;

for rep=1:35
```

```
    if rep>=1&&rep<=5
        normtzfin=mtzfinL(:,1:end)./max(mtzfinL(:,1:end));
    elseif rep>=6&&rep<=10
        normtzfin=mtzfinF(:,1:end)./max(mtzfinF(:,1:end));
    elseif rep>=11&&rep<=15
        normtzfin=mtzfinS(:,1:end)./max(mtzfinS(:,1:end));
    elseif rep>=16&&rep<=20
        normtzfin=mtzfinG(:,1:end)./max(mtzfinG(:,1:end));
    elseif rep>=21&&rep<=25
        normtzfin=mtzfinAf(:,1:end)./max(mtzfinAf(:,1:end));
    elseif rep>=26&&rep<=30
        normtzfin=mtzfinAel(:,1:end)./max(mtzfinAel(:,1:end));
    elseif rep>=31&&rep<=35
        normtzfin=mtzfinAest(:,1:end)./max(mtzfinAest(:,1:end));
    end

pos = randperm(size(normtzfin,1));
normtzfin=normtzfin(pos, :);
dataTrain=normtzfin(1:0.75*size(normtzfin, 1), :);
XTrain = dataTrain(:, 1:end-1);
YTrain = dataTrain(:, end);
dataTest=normtzfin(0.75*size(normtzfin,1)+1:end, :);
XTest = dataTest(:, 1:end-1);
YTest = dataTest(:, end);

for evIP = 1:length(evalIMUposition)
indIMUs = evalIMUposition(evIP);
XTraino=XTrain(:, [indIMUs{1,1}]);
XTesto=XTest(:, [indIMUs{1,1}]);

[err_NB_Train,err_NB_Test,err_NB_Train_kernel,err_NB_Test_kernel,...
err_SVM_Train,err_SVM_Test,err_SVM_lin_Train,err_SVM_lin_Test,...
err_SVM_gauss_Train,err_SVM_gauss_Test,err_SVM_pol_Train,...
err_SVM_pol_Test,err_RF_Train,err_RF_Test,err_KNN_Train,...
err_KNN_Test,CMNB,orderNB,CMNB_k,orderNB_k,CMSVM,orderSVM,CMSVM1,...
orderSVM1,CMSVMg,orderSVMg,CMSVMp,orderSVMp,CMRF,orderRF,CMKNN,...
orderKNN]=clasificadores_bin(XTraino, YTrain, XTesto, YTest);

ConfusionMatNB{evIP}=CMNB;
orderMatNB{evIP}=orderNB;
ConfusionMatNB_k{evIP}=CMNB_k;
orderMatNB_k{evIP}=orderNB_k;
ConfusionMatSVM{evIP}=CMSVM;
orderMatSVM{evIP}=orderSVM;
ConfusionMatSVM1{evIP}=CMSVM1;
```

```
orderMatSVMl{evIP}=orderSVMl;
ConfusionMatSVMg{evIP}=CMSVMg;
orderMatSVMg{evIP}=orderSVMg;
ConfusionMatSVMp{evIP}=CMSVMp;
orderMatSVMp{evIP}=orderSVMp;
ConfusionMatRF{evIP}=CMRF;
orderMatRF{evIP}=orderRF;
ConfusionMatKNN{evIP}=CMKNN;
orderMatKNN{evIP}=orderKNN;

ConfusionMat=[ConfusionMatNB;ConfusionMatNB_k;ConfusionMatSVM;...
    ConfusionMatSVMl;ConfusionMatSVMg;ConfusionMatSVMp;...
    ConfusionMatRF;ConfusionMatKNN];

orderMat=[orderMatNB;orderMatNB_k;orderMatSVM;orderMatSVMl;orderMatSVMg;...
    orderMatSVMp;orderMatRF;orderMatKNN];

erroresTEST{:,evIP} = {err_NB_Test; err_NB_Test_kernel; err_SVM_Test;...
    err_SVM_lin_Test; err_SVM_gauss_Test; err_SVM_pol_Test;err_RF_Test;
        err_KNN_Test};

erroresTRAIN{:,evIP} = {err_NB_Train; err_NB_Train_kernel; err_SVM_Train;...
    err_SVM_lin_Train; err_SVM_gauss_Train; err_SVM_pol_Train; err_RF_Train;
        err_KNN_Train};
end
save(['errores_bin_' num2str(rep)], 'erroresTEST', 'erroresTRAIN', 'XTrain', '
    YTrain', 'XTest', 'YTest','ConfusionMat', 'orderMat');
end
```

E Código para la identificación de los ejercicios bien hechos

```
imuRS = [1:12 25:36];
imuRT = [13:24 37:48];
imuLS = [49:60 73:84];
imuLT = [61:72 85:96];


evalIMUposition={imuLT,imuLS,imuRT,imuRS,[imuLS imuLT],[imuLT imuRT],...
    [imuLT imuRS],[imuLS imuRT],[imuLS imuRS],[imuRS imuRT],...
    [imuLS imuLT imuRT],[imuLS imuRS imuRT],[imuLT imuRS imuRT],...
    [imuLS imuLT imuRS],[imuLS imuLT imuRS imuRT]};


exercises_mcls0;
```

```
for rep=1:5

% elevacion lateral de pierna
% bien
mtzfinL=mtzfinLo(:, 1:size(mtzfinLo,2) -1);
posL = randperm(size(mtzfinL,1));
mtzfinL=mtzfinL(posL, :);
labelsL=labelsLo(posL, :);


% flexo-extension de pierna
% bien
mtzfinF=mtzfinFo(:, 1:size(mtzfinFo,2) -1);
posF = randperm(size(mtzfinF,1));
mtzfinF=mtzfinF(posF, :);
labelsF=labelsFo(posF, :);


% sentadilla
% bien
mtzfinS=mtzfinSo(:, 1:size(mtzfinSo,2) -1);
posS = randperm(size(mtzfinS,1));
mtzfinS=mtzfinS(posS, :);
labelsS=labelsSo(posS, :);


% caminar
% bien
mtzfinG=mtzfinGo(:, 1:size(mtzfinGo,2) -1);
posG = randperm(size(mtzfinG,1));
mtzfinG=mtzfinG(posG, :);
labelsG=labelsGo(posG, :);



% flexo-extension de brazo
% bien
mtzfinAf=mtzfinAfo(:, 1:size(mtzfinAfo,2) -1);
posAf = randperm(size(mtzfinAf,1));
mtzfinAf=mtzfinAf(posAf, :);
labelsAf=labelsAfo(posAf, :);
```

```
% Elevacion de brazos por encima de la cabeza
% bien
mtzfinAel=mtzfinAelo(:, 1:size(mtzfinAelo,2) -1);
posAel = randperm(size(mtzfinAel,1));
mtzfinAel=mtzfinAel(posAel, :);
labelsAel=labelsAelo(posAel, :);


% Estrujar
% bien
mtzfinAest=mtzfinAesto(:, 1:size(mtzfinAesto,2) -1);
posAest = randperm(size(mtzfinAest,1));
mtzfinAest=mtzfinAest(posAest, :);
labelsAest=labelsAesto(posAest, :);


%matriz de ceros
ult=min([size(mtzfinG,1) size(mtzfinS,1) size(mtzfinF,1) size(mtzfinL,1)...
    size(mtzfinAf,1) size(mtzfinAel,1) size(mtzfinAest,1)]);
matriz=[mtzfinG(1:ult, :); mtzfinS(1:ult, :); mtzfinF(1:ult, :);...
    mtzfinL(1:ult, :); mtzfinAf(1:ult, :); mtzfinAel(1:ult, :); ...
    mtzfinAest(1:ult, :)];
labels=[labelsG(1:ult, :); labelsS(1:ult, :); labelsF(1:ult, :);...
    labelsL(1:ult, :); labelsAf(1:ult, :); labelsAel(1:ult, :); ...
    labelsAest(1:ult, :)];
normtz=matriz(:,1:end)./max(matriz(:,1:end));
pos = randperm(size(normtz,1));
matriz=normtz(pos, :);
labels=labels(pos, :);


XTrain_total=matriz(1:0.75*size(matriz, 1), :);
YTrain_total=labels(1:0.75*size(matriz, 1), :);

XTest_total=matriz(0.75*size(matriz,1)+1:end, :);
YTest_total=labels(0.75*size(matriz,1)+1:end, :);


for evIP = 1:length(evalIMUposition)
indIMUs = evalIMUposition(evIP);

XTrain=XTrain_total(:, [indIMUs{1,1}]);
XTest=XTest_total(:, [indIMUs{1,1}]);

[err_NB_Train,err_NB_Test,err_NB_Train_kernel,err_NB_Test_kernel,...
```

```
err_SVM_Train,err_SVM_Test,err_SVM_lin_Train,err_SVM_lin_Test,...
err_SVM_gauss_Train,err_SVM_gauss_Test,err_SVM_pol_Train,...
err_SVM_pol_Test,err_RF_Train,err_RF_Test,err_KNN_Train,...
err_KNN_Test,CMNB,orderNB,CMNB_k,orderNB_k,CMSVM,orderSVM,CMSVMl,...
orderSVMl,CMSVMg,orderSVMg,CMSVMp,orderSVMp,CMRF,orderRF,CMKNN,...
orderKNN]=clasificadores(XTrain, YTrain_total, XTest, YTest_total);

ConfusionMatNB{evIP}=CMNB;
orderMatNB{evIP}=orderNB;
ConfusionMatNB_k{evIP}=CMNB_k;
orderMatNB_k{evIP}=orderNB_k;
ConfusionMatSVM{evIP}=CMSVM;
orderMatSVM{evIP}=orderSVM;
ConfusionMatSVMl{evIP}=CMSVMl;
orderMatSVMl{evIP}=orderSVMl;
ConfusionMatSVMg{evIP}=CMSVMg;
orderMatSVMg{evIP}=orderSVMg;
ConfusionMatSVMp{evIP}=CMSVMp;
orderMatSVMp{evIP}=orderSVMp;
ConfusionMatRF{evIP}=CMRF;
orderMatRF{evIP}=orderRF;
ConfusionMatKNN{evIP}=CMKNN;
orderMatKNN{evIP}=orderKNN;

ConfusionMat=[ConfusionMatNB;ConfusionMatNB_k;ConfusionMatSVM;...
    ConfusionMatSVMl;ConfusionMatSVMg;ConfusionMatSVMp;...
    ConfusionMatRF;ConfusionMatKNN];

orderMat=[orderMatNB;orderMatNB_k;orderMatSVM;orderMatSVMl;orderMatSVMg;...
    orderMatSVMp;orderMatRF;orderMatKNN];

erroresTEST{:,evIP} = {err_NB_Test; err_NB_Test_kernel; err_SVM_Test;...
    err_SVM_lin_Test; err_SVM_gauss_Test; err_SVM_pol_Test;err_RF_Test;
        err_KNN_Test};

erroresTRAIN{:,evIP} = {err_NB_Train; err_NB_Train_kernel; err_SVM_Train;...
    err_SVM_lin_Train; err_SVM_gauss_Train; err_SVM_pol_Train; err_RF_Train;
        err_KNN_Train};

end

save(['errores_multiclase_0_rep' num2str(rep)],'erroresTEST','erroresTRAIN','
    XTrain_total','YTrain_total','XTest_total','YTest_total','ConfusionMat','
    orderMat');
```

```
end
```

F Código para la identificación de los ejercicios bien y mal hechos

```
imuRS = [1:12  25:36];
imuRT = [13:24  37:48];
imuLS = [49:60  73:84];
imuLT = [61:72  85:96];


evalIMUposition={imuLT,imuLS,imuRT,imuRS,...
        [imuLS imuLT],[imuLT imuRT],[imuLT imuRS],[imuLS imuRT],[imuLS imuRS],[
            imuRS imuRT],...
        [imuLS imuLT imuRT],[imuLS imuRS imuRT],[imuLT imuRS imuRT],[imuLS
            imuLT imuRS],...
        [imuLS imuLT imuRS imuRT]};



exercises_mcls01_7classes


for rep=1:10

%  elevacion lateral de pierna
mtzfinL=mtzfinLo(:, 1:size(mtzfinLo,2) -1);
posL = randperm(size(mtzfinL,1));
mtzfinL=mtzfinL(posL, :);
labelsL=labelsLo(posL, :);

%  flexo-extension de pierna
mtzfinF=mtzfinFo(:, 1:size(mtzfinFo,2) -1);
posF = randperm(size(mtzfinF,1));
mtzfinF=mtzfinF(posF, :);
labelsF=labelsFo(posF, :);

%  sentadilla
mtzfinS=mtzfinSo(:, 1:size(mtzfinSo,2) -1);
posS = randperm(size(mtzfinS,1));
mtzfinS=mtzfinS(posS, :);
labelsS=labelsSo(posS, :);
```

```
% caminar
mtzfinG=mtzfinGo(:, 1:size(mtzfinGo,2) -1);
posG = randperm(size(mtzfinG,1));
mtzfinG=mtzfinG(posG, :);
labelsG=labelsGo(posG, :);

% flexo-extension de brazo
mtzfinAf=mtzfinAfo(:, 1:size(mtzfinAfo,2) -1);
posAf = randperm(size(mtzfinAf,1));
mtzfinAf=mtzfinAf(posAf, :);
labelsAf=labelsAfo(posAf, :);

% elevacion de brazos por encima de la cabeza
mtzfinAel=mtzfinAelo(:, 1:size(mtzfinAelo,2) -1);
posAel = randperm(size(mtzfinAel,1));
mtzfinAel=mtzfinAel(posAel, :);
labelsAel=labelsAelo(posAel, :);

% estrujar
mtzfinAest=mtzfinAesto(:, 1:size(mtzfinAesto,2) -1);
posAest = randperm(size(mtzfinAest,1));
mtzfinAest=mtzfinAest(posAest, :);
labelsAest=labelsAesto(posAest, :);



% matriz
ult=min([size(mtzfinG,1) size(mtzfinS,1) size(mtzfinF,1) size(mtzfinL,1)...
    size(mtzfinAf,1) size(mtzfinAel,1) size(mtzfinAest,1)]);
matriz=[mtzfinG(1:ult, :); mtzfinS(1:ult, :); mtzfinF(1:ult, :);...
    mtzfinL(1:ult, :); mtzfinAf(1:ult, :); mtzfinAel(1:ult, :); ...
    mtzfinAest(1:ult, :)];
labels=[labelsG(1:ult, :); labelsS(1:ult, :); labelsF(1:ult, :);...
    labelsL(1:ult, :); labelsAf(1:ult, :); labelsAel(1:ult, :); ...
    labelsAest(1:ult, :)];
normtz=matriz(:,1:end)./max(matriz(:,1:end));
pos = randperm(size(normtz,1));
matriz=normtz(pos, :);
labels=labels(pos, :);

XTrain_total=matriz(1:0.75*size(matriz, 1), :);
YTrain_total=labels(1:0.75*size(matriz, 1), :);

XTest_total=matriz(0.75*size(matriz,1)+1:end, :);
YTest_total=labels(0.75*size(matriz,1)+1:end, :);
```

```
for evIP = 1:length(evalIMUposition)
indIMUs = evalIMUposition(evIP);

XTrain=XTrain_total(:, [indIMUs{1,1}]);
XTest=XTest_total(:, [indIMUs{1,1}]);

[err_NB_Train,err_NB_Test,err_NB_Train_kernel,err_NB_Test_kernel,...
err_SVM_Train,err_SVM_Test,err_SVM_lin_Train,err_SVM_lin_Test,...
err_SVM_gauss_Train,err_SVM_gauss_Test,err_SVM_pol_Train,...
err_SVM_pol_Test,err_RF_Train,err_RF_Test,err_KNN_Train,...
err_KNN_Test,CMNB,orderNB,CMNB_k,orderNB_k,CMSVM,orderSVM,CMSVMl,...
orderSVMl,CMSVMg,orderSVMg,CMSVMp,orderSVMp,CMRF,orderRF,CMKNN,...
orderKNN]=clasificadores(XTrain, YTrain_total, XTest, YTest_total);

ConfusionMatNB{evIP}=CMNB;
orderMatNB{evIP}=orderNB;
ConfusionMatNB_k{evIP}=CMNB_k;
orderMatNB_k{evIP}=orderNB_k;
ConfusionMatSVM{evIP}=CMSVM;
orderMatSVM{evIP}=orderSVM;
ConfusionMatSVMl{evIP}=CMSVMl;
orderMatSVMl{evIP}=orderSVMl;
ConfusionMatSVMg{evIP}=CMSVMg;
orderMatSVMg{evIP}=orderSVMg;
ConfusionMatSVMp{evIP}=CMSVMp;
orderMatSVMp{evIP}=orderSVMp;
ConfusionMatRF{evIP}=CMRF;
orderMatRF{evIP}=orderRF;
ConfusionMatKNN{evIP}=CMKNN;
orderMatKNN{evIP}=orderKNN;

ConfusionMat=[ConfusionMatNB;ConfusionMatNB_k;ConfusionMatSVM;...
    ConfusionMatSVMl;ConfusionMatSVMg;ConfusionMatSVMp;...
    ConfusionMatRF;ConfusionMatKNN];

orderMat=[orderMatNB;orderMatNB_k;orderMatSVM;orderMatSVMl;orderMatSVMg;...
    orderMatSVMp;orderMatRF;orderMatKNN];

erroresTEST{:,evIP} = {err_NB_Test; err_NB_Test_kernel; err_SVM_Test;...
    err_SVM_lin_Test; err_SVM_gauss_Test; err_SVM_pol_Test;err_RF_Test;
        err_KNN_Test};

erroresTRAIN{:,evIP} = {err_NB_Train; err_NB_Train_kernel; err_SVM_Train;...
```

```
        err_SVM_lin_Train; err_SVM_gauss_Train; err_SVM_pol_Train; err_RF_Train;
            err_KNN_Train};

    end
    save(['errores_multiclase_0_1_7_classes_rep' num2str(rep)],'erroresTEST','
        erroresTRAIN','XTrain_total','YTrain_total','XTest_total','YTest_total','
        ConfusionMat','orderMat');
    end
```

G Código para la evaluación e identificación entre los ejercicios bien hechos y una clase denominada *wrng*

```
imuRS = [1:12  25:36];
imuRT = [13:24  37:48];
imuLS = [49:60  73:84];
imuLT = [61:72  85:96];


evalIMUposition={imuLT,imuLS,imuRT,imuRS,[imuLS imuLT],[imuLT imuRT],...
    [imuLT imuRS],[imuLS imuRT],[imuLS imuRS],[imuRS imuRT],...
    [imuLS imuLT imuRT],[imuLS imuRS imuRT],[imuLT imuRS imuRT],...
    [imuLS imuLT imuRS],[imuLS imuLT imuRS imuRT]};

exercises_mcls0;
exercises_mcls1;


for rep=1:5


% elevacion lateral de pierna
% bien
mtzfinL=mtzfinLo(:, 1:size(mtzfinLo,2) -1);
posL = randperm(size(mtzfinL,1));
mtzfinL=mtzfinL(posL, :);
labelsL=labelsLo(posL, :);
% mal
mtzfinL1=mtzfinL1o(:, 1:size(mtzfinL1o,2) -1);
posL1 = randperm(size(mtzfinL1,1));
mtzfinL1=mtzfinL1(posL1, :);
labelsL1=labelsL1o(posL1, :);
```

```
% flexo-extension de pierna
% bien
mtzfinF=mtzfinFo(:, 1:size(mtzfinFo,2) -1);
posF = randperm(size(mtzfinF,1));
mtzfinF=mtzfinF(posF, :);
labelsF=labelsFo(posF, :);
% mal
mtzfinF1=mtzfinF1o(:, 1:size(mtzfinF1o,2) -1);
posF1 = randperm(size(mtzfinF1,1));
mtzfinF1=mtzfinF1(posF1, :);
labelsF1=labelsF1o(posF1, :);


% sentadilla
% bien
mtzfinS=mtzfinSo(:, 1:size(mtzfinSo,2) -1);
posS = randperm(size(mtzfinS,1));
mtzfinS=mtzfinS(posS, :);
labelsS=labelsSo(posS, :);
% mal
mtzfinS1=mtzfinS1o(:, 1:size(mtzfinS1o,2) -1);
posS1 = randperm(size(mtzfinS1,1));
mtzfinS1=mtzfinS1(posS1, :);
labelsS1=labelsS1o(posS1, :);


% caminar
% bien
mtzfinG=mtzfinGo(:, 1:size(mtzfinGo,2) -1);
posG = randperm(size(mtzfinG,1));
mtzfinG=mtzfinG(posG, :);
labelsG=labelsGo(posG, :);
% mal
mtzfinG1=mtzfinG1o(:, 1:size(mtzfinG1o,2) -1);
posG1 = randperm(size(mtzfinG1,1));
mtzfinG1=mtzfinG1(posG1, :);
labelsG1=labelsG1o(posG1, :);


% flexo-extension de brazo
% bien
mtzfinAf=mtzfinAfo(:, 1:size(mtzfinAfo,2) -1);
posAf = randperm(size(mtzfinAf,1));
mtzfinAf=mtzfinAf(posAf, :);
labelsAf=labelsAfo(posAf, :);
```

```
% mal
mtzfinAf1=mtzfinAf1o(:, 1:size(mtzfinAf1o,2) -1);
posAf1 = randperm(size(mtzfinAf1,1));
mtzfinAf1=mtzfinAf1(posAf1, :);
labelsAf1=labelsAf1o(posAf1, :);


% Elevacion de brazos por encima de la cabeza
% bien
mtzfinAel=mtzfinAelo(:, 1:size(mtzfinAelo,2) -1);
posAel = randperm(size(mtzfinAel,1));
mtzfinAel=mtzfinAel(posAel, :);
labelsAel=labelsAelo(posAel, :);
% mal
mtzfinAel1=mtzfinAel1o(:, 1:size(mtzfinAel1o,2) -1);
posAel1 = randperm(size(mtzfinAel1,1));
mtzfinAel1=mtzfinAel1(posAel1, :);
labelsAel1=labelsAel1o(posAel1, :);


% Estrujar
% bien
mtzfinAest=mtzfinAesto(:, 1:size(mtzfinAesto,2) -1);
posAest = randperm(size(mtzfinAest,1));
mtzfinAest=mtzfinAest(posAest, :);
labelsAest=labelsAesto(posAest, :);
% mal
mtzfinAest1=mtzfinAest1o(:, 1:size(mtzfinAest1o,2) -1);
posAest1 = randperm(size(mtzfinAest1,1));
mtzfinAest1=mtzfinAest1(posAest1, :);
labelsAest1=labelsAest1o(posAest1, :);


%matriz de ceros
ult=min([size(mtzfinG,1) size(mtzfinS,1) size(mtzfinF,1) size(mtzfinL,1)...
    size(mtzfinAf,1) size(mtzfinAel,1) size(mtzfinAest,1)]);
matriz0=[mtzfinG(1:ult, :); mtzfinS(1:ult, :); mtzfinF(1:ult, :);...
    mtzfinL(1:ult, :); mtzfinAf(1:ult, :); mtzfinAel(1:ult, :); ...
    mtzfinAest(1:ult, :)];
labels0=[labelsG(1:ult, :); labelsS(1:ult, :); labelsF(1:ult, :);...
    labelsL(1:ult, :); labelsAf(1:ult, :); labelsAel(1:ult, :); ...
    labelsAest(1:ult, :)];

% matriz de unos
mtz1=[mtzfinG1; mtzfinS1; mtzfinF1; mtzfinL1; mtzfinAf1; mtzfinAel1; ...
```

```
    mtzfinAest1];
labels1_x(:,1)=char('w'*ones(length(mtz1), 1));
labels1_x(:,2)=char('r'*ones(length(mtz1), 1));
labels1_x(:,3)=char('n'*ones(length(mtz1), 1));
labels1_x(:,4)=char('g'*ones(length(mtz1), 1));
matriz1=mtz1(1:ult,:);
labels1=labels1_x(1:ult,:);

% matriz de ceros y unos
matriz=[matriz0; matriz1];
labels=[labels0; labels1];
normtz=matriz(:,1:end)./max(matriz(:,1:end));
pos = randperm(size(normtz,1));
matriz=normtz(pos, :);
labels=labels(pos, :);


XTrain_total=matriz(1:0.75*size(matriz, 1), :);
YTrain_total=labels(1:0.75*size(matriz, 1), :);

XTest_total=matriz(0.75*size(matriz,1)+1:end, :);
YTest_total=labels(0.75*size(matriz,1)+1:end, :);


for evIP = 1:length(evalIMUposition)
indIMUs = evalIMUposition(evIP);

XTrain=XTrain_total(:, [indIMUs{1,1}]);
XTest=XTest_total(:, [indIMUs{1,1}]);

[err_NB_Train,err_NB_Test,err_NB_Train_kernel,err_NB_Test_kernel,...
err_SVM_Train,err_SVM_Test,err_SVM_lin_Train,err_SVM_lin_Test,...
err_SVM_gauss_Train,err_SVM_gauss_Test,err_SVM_pol_Train,...
err_SVM_pol_Test,err_RF_Train,err_RF_Test,err_KNN_Train,...
err_KNN_Test,CMNB,orderNB,CMNB_k,orderNB_k,CMSVM,orderSVM,CMSVM1,...
orderSVM1,CMSVMg,orderSVMg,CMSVMp,orderSVMp,CMRF,orderRF,CMKNN,...
orderKNN]=clasificadores(XTrain, YTrain_total, XTest, YTest_total);

ConfusionMatNB{evIP}=CMNB;
orderMatNB{evIP}=orderNB;
ConfusionMatNB_k{evIP}=CMNB_k;
orderMatNB_k{evIP}=orderNB_k;
ConfusionMatSVM{evIP}=CMSVM;
orderMatSVM{evIP}=orderSVM;
ConfusionMatSVM1{evIP}=CMSVM1;
```

```
orderMatSVMl{evIP}=orderSVMl;
ConfusionMatSVMg{evIP}=CMSVMg;
orderMatSVMg{evIP}=orderSVMg;
ConfusionMatSVMp{evIP}=CMSVMp;
orderMatSVMp{evIP}=orderSVMp;
ConfusionMatRF{evIP}=CMRF;
orderMatRF{evIP}=orderRF;
ConfusionMatKNN{evIP}=CMKNN;
orderMatKNN{evIP}=orderKNN;

ConfusionMat=[ConfusionMatNB;ConfusionMatNB_k;ConfusionMatSVM;...
    ConfusionMatSVMl;ConfusionMatSVMg;ConfusionMatSVMp;...
    ConfusionMatRF;ConfusionMatKNN];

orderMat=[orderMatNB;orderMatNB_k;orderMatSVM;orderMatSVMl;orderMatSVMg;...
    orderMatSVMp;orderMatRF;orderMatKNN];

erroresTEST{:,evIP} = {err_NB_Test; err_NB_Test_kernel; err_SVM_Test;...
    err_SVM_lin_Test; err_SVM_gauss_Test; err_SVM_pol_Test;err_RF_Test;
        err_KNN_Test};

erroresTRAIN{:,evIP} = {err_NB_Train; err_NB_Train_kernel; err_SVM_Train;...
    err_SVM_lin_Train; err_SVM_gauss_Train; err_SVM_pol_Train; err_RF_Train;
        err_KNN_Train};

end

save(['errores_multiclase_0_1_rep' num2str(rep)],'erroresTEST','erroresTRAIN',
    'XTrain_total','YTrain_total','XTest_total','YTest_total','ConfusionMat','
    orderMat');

end
```

H Código para la evaluación e identificación de los ejercicios bien y mal hechos

```
imuRS = [1:12  25:36];
imuRT = [13:24  37:48];
imuLS = [49:60  73:84];
imuLT = [61:72  85:96];


evalIMUposition={imuLT,imuLS,imuRT,imuRS,...
```

```
        [imuLS imuLT],[imuLT imuRT],[imuLT imuRS],[imuLS imuRT],[imuLS imuRS],[
            imuRS imuRT],...
        [imuLS imuLT imuRT],[imuLS imuRS imuRT],[imuLT imuRS imuRT],[imuLS
            imuLT imuRS],...
        [imuLS imuLT imuRS imuRT]};



exercises_mcls0;
exercises_mcls1;


for rep=1:10

%  Elevacion lateral de pierna
% bien
mtzfinL=mtzfinLo(:, 1:size(mtzfinLo,2) -1);
posL = randperm(size(mtzfinL,1));
mtzfinL=mtzfinL(posL, :);
labelsL=labelsLo(posL, :);
% mal
mtzfinL1=mtzfinL1o(:, 1:size(mtzfinL1o,2) -1);
posL1 = randperm(size(mtzfinL1,1));
mtzfinL1=mtzfinL1(posL1, :);
labelsL1=labelsL1o(posL1, :);

%  Flexo-extension de pierna
% bien
mtzfinF=mtzfinFo(:, 1:size(mtzfinFo,2) -1);
posF = randperm(size(mtzfinF,1));
mtzfinF=mtzfinF(posF, :);
labelsF=labelsFo(posF, :);
% mal
mtzfinF1=mtzfinF1o(:, 1:size(mtzfinF1o,2) -1);
posF1 = randperm(size(mtzfinF1,1));
mtzfinF1=mtzfinF1(posF1, :);
labelsF1=labelsF1o(posF1, :);

%  Sentadilla
% bien
mtzfinS=mtzfinSo(:, 1:size(mtzfinSo,2) -1);
posS = randperm(size(mtzfinS,1));
mtzfinS=mtzfinS(posS, :);
labelsS=labelsSo(posS, :);
% mal
```

```
mtzfinS1=mtzfinS1o(:, 1:size(mtzfinS1o,2) -1);
posS1 = randperm(size(mtzfinS1,1));
mtzfinS1=mtzfinS1(posS1, :);
labelsS1=labelsS1o(posS1, :);

%  Caminar
% bien
mtzfinG=mtzfinGo(:, 1:size(mtzfinGo,2) -1);
posG = randperm(size(mtzfinG,1));
mtzfinG=mtzfinG(posG, :);
labelsG=labelsGo(posG, :);
% mal
mtzfinG1=mtzfinG1o(:, 1:size(mtzfinG1o,2) -1);
posG1 = randperm(size(mtzfinG1,1));
mtzfinG1=mtzfinG1(posG1, :);
labelsG1=labelsG1o(posG1, :);

%  Flexo-extension de brazo
% bien
mtzfinAf=mtzfinAfo(:, 1:size(mtzfinAfo,2) -1);
posAf = randperm(size(mtzfinAf,1));
mtzfinAf=mtzfinAf(posAf, :);
labelsAf=labelsAfo(posAf, :);
% mal
mtzfinAf1=mtzfinAf1o(:, 1:size(mtzfinAf1o,2) -1);
posAf1 = randperm(size(mtzfinAf1,1));
mtzfinAf1=mtzfinAf1(posAf1, :);
labelsAf1=labelsAf1o(posAf1, :);

%  Elevacion de brazos por encima de la cabeza
% bien
mtzfinAel=mtzfinAelo(:, 1:size(mtzfinAelo,2) -1);
posAel = randperm(size(mtzfinAel,1));
mtzfinAel=mtzfinAel(posAel, :);
labelsAel=labelsAelo(posAel, :);
% mal
mtzfinAel1=mtzfinAel1o(:, 1:size(mtzfinAel1o,2) -1);
posAel1 = randperm(size(mtzfinAel1,1));
mtzfinAel1=mtzfinAel1(posAel1, :);
labelsAel1=labelsAel1o(posAel1, :);

%  Estrujar
% bien
mtzfinAest=mtzfinAesto(:, 1:size(mtzfinAesto,2) -1);
posAest = randperm(size(mtzfinAest,1));
```

```
mtzfinAest=mtzfinAest(posAest, :);
labelsAest=labelsAesto(posAest, :);
% mal
mtzfinAest1=mtzfinAest1o(:, 1:size(mtzfinAest1o,2) -1);
posAest1 = randperm(size(mtzfinAest1,1));
mtzfinAest1=mtzfinAest1(posAest1, :);
labelsAest1=labelsAest1o(posAest1, :);


%matriz de ceros
ult=min([size(mtzfinG,1) size(mtzfinS,1) size(mtzfinF,1) size(mtzfinL,1)...
    size(mtzfinAf,1) size(mtzfinAel,1) size(mtzfinAest,1)]);
matriz0=[mtzfinG(1:ult, :); mtzfinS(1:ult, :); mtzfinF(1:ult, :);...
    mtzfinL(1:ult, :); mtzfinAf(1:ult, :); mtzfinAel(1:ult, :); ...
    mtzfinAest(1:ult, :)];
labels0=[labelsG(1:ult, :); labelsS(1:ult, :); labelsF(1:ult, :);...
    labelsL(1:ult, :); labelsAf(1:ult, :); labelsAel(1:ult, :); ...
    labelsAest(1:ult, :)];

% matriz de unos
ult1=min([size(mtzfinG1,1) size(mtzfinS1,1) size(mtzfinF1,1) size(mtzfinL1,1)
    ...
    size(mtzfinAf1,1) size(mtzfinAel1,1) size(mtzfinAest1,1)]);
matriz1=[mtzfinG1(1:ult1, :); mtzfinS1(1:ult1, :); mtzfinF1(1:ult1, :);...
    mtzfinL1(1:ult1, :); mtzfinAf1(1:ult1, :); mtzfinAel1(1:ult1, :); ...
    mtzfinAest1(1:ult1, :)];
labels1=[labelsG1(1:ult1, :); labelsS1(1:ult1, :); labelsF1(1:ult1, :);...
    labelsL1(1:ult1, :); labelsAf1(1:ult1, :); labelsAel1(1:ult1, :); ...
    labelsAest1(1:ult1, :)];

% matriz de ceros y unos
matriz=[matriz0; matriz1];
labels=[labels0; labels1];
normtz=matriz(:,1:end)./max(matriz(:,1:end));
pos = randperm(size(normtz,1));
matriz=normtz(pos, :);
labels=labels(pos, :);



XTrain_total=matriz(1:0.75*size(matriz, 1), :);
YTrain_total=labels(1:0.75*size(matriz, 1), :);

XTest_total=matriz(0.75*size(matriz,1)+1:end, :);
YTest_total=labels(0.75*size(matriz,1)+1:end, :);
```

```
for evIP = 1:length(evalIMUposition)
indIMUs = evalIMUposition(evIP);

XTrain=XTrain_total(:, [indIMUs{1,1}]);
XTest=XTest_total(:, [indIMUs{1,1}]);

    [err_NB_Train,err_NB_Test,err_NB_Train_kernel,err_NB_Test_kernel,...
err_SVM_Train,err_SVM_Test,err_SVM_lin_Train,err_SVM_lin_Test,...
err_SVM_gauss_Train,err_SVM_gauss_Test,err_SVM_pol_Train,...
err_SVM_pol_Test,err_RF_Train,err_RF_Test,err_KNN_Train,...
err_KNN_Test,CMNB,orderNB,CMNB_k,orderNB_k,CMSVM,orderSVM,CMSVMl,...
orderSVMl,CMSVMg,orderSVMg,CMSVMp,orderSVMp,CMRF,orderRF,CMKNN,...
orderKNN]=clasificadores(XTrain, YTrain_total, XTest, YTest_total);

ConfusionMatNB{evIP}=CMNB;
orderMatNB{evIP}=orderNB;
ConfusionMatNB_k{evIP}=CMNB_k;
orderMatNB_k{evIP}=orderNB_k;
ConfusionMatSVM{evIP}=CMSVM;
orderMatSVM{evIP}=orderSVM;
ConfusionMatSVMl{evIP}=CMSVMl;
orderMatSVMl{evIP}=orderSVMl;
ConfusionMatSVMg{evIP}=CMSVMg;
orderMatSVMg{evIP}=orderSVMg;
ConfusionMatSVMp{evIP}=CMSVMp;
orderMatSVMp{evIP}=orderSVMp;
ConfusionMatRF{evIP}=CMRF;
orderMatRF{evIP}=orderRF;
ConfusionMatKNN{evIP}=CMKNN;
orderMatKNN{evIP}=orderKNN;

ConfusionMat=[ConfusionMatNB;ConfusionMatNB_k;ConfusionMatSVM;...
    ConfusionMatSVMl;ConfusionMatSVMg;ConfusionMatSVMp;...
    ConfusionMatRF;ConfusionMatKNN];

orderMat=[orderMatNB;orderMatNB_k;orderMatSVM;orderMatSVMl;orderMatSVMg;...
    orderMatSVMp;orderMatRF;orderMatKNN];

erroresTEST{:,evIP} = {err_NB_Test; err_NB_Test_kernel; err_SVM_Test;...
    err_SVM_lin_Test; err_SVM_gauss_Test; err_SVM_pol_Test;err_RF_Test;
        err_KNN_Test};

erroresTRAIN{:,evIP} = {err_NB_Train; err_NB_Train_kernel; err_SVM_Train;...
```

```
        err_SVM_lin_Train; err_SVM_gauss_Train; err_SVM_pol_Train; err_RF_Train;
            err_KNN_Train};

    end
    save(['errores_multiclase_0_1_14_classes_rep' num2str(rep)],'erroresTEST','
        erroresTRAIN','XTrain_total','YTrain_total','XTest_total','YTest_total','
        ConfusionMat','orderMat');
    end
```

Bibliografía

[1] Javier Conte Alcaraz, Sanam Moghaddamnia, and Jürgen Peissig. Mobile quantification and therapy course tracking for gait rehabilitation. *arXiv e-prints*, pages arXiv–1707, 2017.

[2] Je-Nam Kim, Woo Suk Chong, Seong-Hyun Kim, and Kyong Kim. Health management monitoring system for use in classifying lower extremity movements of the elderly. *Journal of Mechanics in Medicine and Biology*, 19(07):1940028, 2019.

[3] Ezio Preatoni, Stefano Nodari, and Nicola Francesco Lopomo. Supervised machine learning applied to wearable sensor data can accurately classify functional fitness exercises within a continuous workout. *Frontiers in Bioengineering and Biotechnology*, 8:664, 2020.

[4] Luckshman Bavan, Karl Surmacz, David Beard, Stephen Mellon, and Jonathan Rees. Adherence monitoring of rehabilitation exercise with inertial sensors: A clinical validation study. *Gait & posture*, 70:211–217, 2019.

[5] Ana Pereira, Duarte Folgado, Ricardo Cotrim, and Inês Sousa. Physiotherapy exercises evaluation using a combined approach based on semg and wearable inertial sensors. In *BIOSIGNALS*, pages 73–82, 2019.

[6] Steven D Hlucny and Domen Novak. Characterizing human box-lifting behavior using wearable inertial motion sensors. *Sensors*, 20(8):2323, 2020.

[7] Darragh Whelan, Martin O'Reilly, Tomás Ward, Eamonn Delahunt, and Brian Caulfield. Evaluating performance of the lunge exercise with multiple and individual inertial measurement units. In *Pervasive Health 2016: 10th EAI International Conference on Pervasive Computing Technologies for Healthcare, Cancun, Mexico, 16-19 May 2016*. ACM, 2016.

[8] Rezvan Kianifar, Alex Lee, Sachin Raina, and Dana Kulic. Classification of squat quality with inertial measurement units in the single leg squat mobility test. *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, 2016-Octob:6273–6276, 2016. ISSN 1557170X. doi: 10.1109/EMBC.2016.7592162.

[9] Jonathan F.S. Lin and Dana Kulić. Human pose recovery using wireless inertial measurement units. *Physiological Measurement*, 33(12):2099–2115, 2012. ISSN 09673334. doi: 10.1088/0967-3334/33/12/2099.

[10] Estefania Munoz Diaz, Dina Bousdar Ahmed, and Susanna Kaiser. A review of indoor localization methods based on inertial sensors. In *Geographical and Fingerprinting Data to Create Systems for Indoor Positioning and Indoor/Outdoor Navigation*, pages 311–333. Elsevier, 2019.

[11] M Izquierdo, A Casas Herrero, F Zambom Ferraresi, N Martinez Velilla, C Alonso Bouzon, and L Rodriguez Mañas. Programa de ejercicio físico multicompetente: Vivifrail, 2017.

[12] x-io Technologies Limited. NGIMU - x-io technologies. `https://x-io.co.uk/ngimu/`. Accessed: 2021-02-08.

[13] Lei Zhao and Wenjing Chen. Detection and recognition of human body posture in motion based on sensor technology. *IEEJ Transactions on Electrical and Electronic Engineering*, 15(5):766–770, 2020.

Universidad de Alcalá
Escuela Politécnica Superior